Universidad Don Bosco.
Facultad de Ingeniería.
Ingeniería Electrónica.

"Modelador en realidad Virtual de patrones de lóbulos de antenas"

Integrantes : Tito Livio Aparicio Montes AM970019.
Marvin Caballero Zelaya CZ940005.

Soyapango, 29 de Noviembre del 2004.

INDICE

1
DEFINICIÓN DEL TEMA

El proyecto esta orientado en la creación de un sistema por medio de un software, que muestre gráficamente el diagrama de radiación en tres dimensiones de una antena especifica.

En la primera parte de este proyecto trabajaremos con la antena dipolo, la cual será simulado su patrón de radiación en forma ideal .También se simulara su patrón de radiación del dipolo cuando esta bajo el efecto de una interferencia.

Para la antena dipolo con y sin interferencia se obtendrá la expresión o ecuación matemática que determine la forma que tendrá diagrama de radiación de la antena, una vez obtenida dicha expresión, el sistema será capaz de simular los lóbulos de la antena, dicha ecuación ira implícita en el programa.

El programa será capaz de mostrar los cambios del diagrama de radiación y otros parámetros de la antena, esto dependiendo de los valores que el usuario le ingrese, en este caso para la antena dipolo sin interferencia será la relación entre la longitud de la antena y la longitud de onda y en el caso que se le agregue interferencia el otro valor a tomar en cuenta será la distancia dipolo e interferencia.

Las herramientas que se utilizaran para la realización de este software son: Lenguaje de programación Visual Basic V6.0, navegador de VRML (Lenguaje Para el Modelado de la Realidad Virtual), e Internet Explorer.

2
JUSTIFICACIÓN DEL TEMA

El desarrollo de este sistema tendrá la importancia:

Primero, servirá de apoyo didáctico muy importante en la Universidad Don Bosco en materias que involucran el conocimiento de antenas tales como "Seguridad y Protección de Circuitos Electrónicos", "Comunicaciones I" y "Propagación y Antenas". Por medio del software el alumno tendrá una mejor perspectiva para apreciar el diagrama de radiación de la antena en tres dimensiones, la dirección donde la antena propaga una mayor potencia, eso en base a que se podrán variar parámetros tales como la relación entre las dimensiones y frecuencia a la que opera la antena. Con la implementación de dicho software los estudiantes serán motivados hacia el estudio de asignaturas, que como la teoría electrodinámica no siempre resultan accesibles por el lenguaje matemático y la abstracción que requieren.

Segundo, permitirá que un diseñador de antenas tenga la facilidad de observar la forma como se distribuye la potencia de una antena transmisora en el espacio, esto le dará una mayor facilidad para deducir la forma como se puede obtener máxima ganancia de un modelo de antena en base a la forma y dirección de los lóbulos.

3
GENERALIDADES

3.1 Objetivo General

- Crear un programa que pueda mostrar el diagrama de radiación de una antena especifica en tres dimensiones mediante VRML, Internet Explorer y Visual Basic.

3.1.1 Objetivos Específicos

- Para el dipolo se mostrara el diagrama de radiación en 3D donde se variara su forma modificando la relación longitud del dipolo (2H) y longitud de onda (λ).

- Para el dipolo con interferencia se mostrara el diagrama de radiación en 3D donde se variara su forma modificando la relación longitud del dipolo(2H) y longitud de onda(λ) y distancia(a) de la interferencia .

- Motivar a los estudiantes con la implementación del software hacia el estudio de asignaturas relacionada a la propagación y antenas que no siempre resultan accesibles por el lenguaje matemático y la abstracción que requieren.

- Que el usuario obtenga conclusiones de la tendencia de la dirección de propagación de potencia del dipolo al variar la relación longitud del dipolo (2H) y longitud de onda (λ).

3.2 Alcances.

- El software será capaz de mostrar en tres dimensiones el diagrama de radiación de una antena Dipolo y un caso de interferencia de la misma antena, para la primera defensa.

- El software será capaz de mostrar en tres dimensiones el diagrama de radiación de una antena Yagi y una antena parabólica para la segunda defensa.

- Para cada tipo de antena que se modele se lograra modificar la forma del diagrama de radiación, variando los parámetros de dimensión y longitud de onda de la antena.

- Se mostrara una animación donde se tendrá una cámara que visualice el contorno del diagrama de radiación.

3.3 Limitaciones.

- En el diagrama de radiación no se tomaran en cuenta efectos de la ionosfera o de reflexión de la tierra.

- No se realizaran demostraciones con antenas físicas del diagrama de radiación de las antenas, ya que se limitara a la representación de dichos lóbulos en 3D en el ámbito de software.

- No se realizaran demostraciones de enlaces de transmisión entre antenas virtuales, por lo que el programa se limitara a la demostración de los patrones de radiación, de una antena a la vez.

- Únicamente para la antena Dipolo se incluirá un caso en que la antena presentara una fuente de interferencia.

- La ecuación característica del diagrama de radiación de una antena específica ira implícita en el programa, el usuario no la podrá ingresar al programa.

.

4
CONSIDERACIONES GENERALES SOBRE ANTENAS.

4.1 Introducción

La antena es un conductor de longitud definida que se coloca al final de una línea de transmisión, y que se encarga de transmitir al ambiente, o irradiar la señal suministrada por el equipo.
Una antena es la parte del sistema transmisor o receptor, que convierte una señal electrónica (conducida por unos cables) en ondas electromagnéticas (propagadas en el espacio) o viceversa. La antena es un elemento de transición entre la zona de onda guiada y la zona de espacio libre, con ciertas características de direccionalidad (filtrado espacial). Existe una gran variedad de tamaños y diseños de antenas, dependiendo de la frecuencia de las señales su utilización y sus parámetros de operación. Funcionalmente, cualquier antena puede emitir o recibir señales. Sin embargo, las antenas que han sido diseñadas para transmisión de alta potencia deben ser capaces de funcionar con grandes cantidades de potencia.
De todos los elementos de una estación, la antena es el que contiene el comportamiento menos predecible, esto es debido a que interacciona fuertemente con todo lo que le rodea. La misión de la antena es radiar la potencia que se le suministra con las características de direccionalidad adecuadas a la aplicación. Por ejemplo, en radiodifusión o comunicaciones móviles se querrá radiar sobre la zona de cobertura de forma omnidireccional, mientras que en radiocomunicaciones fijas interesara que las antenas sean direccionales.
Existen pues dos misiones básicas de una antena: transmitir y recibir, imponiendo cada aplicación condiciones particulares sobre la direccionalidad de la antena, niveles de potencia que debe soportar, frecuencia de trabajo y otros parámetros. Esta diversidad de situaciones da origen a un gran número de tipos de antena.
Debido a esto y a la necesidad de aprovechar al máximo la señal que emiten nuestros equipos, es importante el estudio de los principios básicos de las antenas, que nos darán una idea general de cómo se comportan, en que condiciones sirve una antena dada, y en que condiciones no es adecuada para esa aplicación.

4.1.1. Bandas de frecuencias de Las antenas de transmisión.

Toda onda se caracteriza por su frecuencia (f) y su longitud de onda (λ), ambas relacionadas por la velocidad de propagación en el medio, que habitualmente en antenas tiene las propiedades del vacío ($c=3x10^8 m/s$). El conjunto de todas las frecuencias, o espectro de frecuencias, se divide por décadas en bandas, con la denominación presentada en la **Tabla 4.1** cada aplicación tiene asignada por los organismos de normalización unas determinadas porciones de ese espectro.

BANDA	FRECUENCIA	LONG. DE ONDA	DENOMINACIÓN
ELF	<3 kHz	>100 km	Frecuencia Extremadamente Baja
VLF	3-30kHz	100-10 km	Frecuencia Muy Baja
LF	30-300 kHz	10-1 km	Baja Frecuencia
MF	0.3-3 MHz	1,000-100 m	Frecuencia Media
HF	3-30 MHz	100–10 m	Alta Frecuencia
VHF	30-300 MHz	10–1 m	Frecuencia Muy Alta
UHF	0.3-3 GHz	100–10 cm	Ultra Alta Frecuencia
SHF	3-30 GHz	10-1 cm	Súper Alta Frecuencia
EHF	30-300 GHz	10-1 mm	Frecuencia Extremadamente Alta

Tabla 4.1 Denominación de las bandas de frecuencias por décadas.

En las frecuencias de microondas existe una subdivisión desde los primeros tiempos del radar, recogida en la **Tabla 4.1.1**, que es ampliamente utilizada en la actualidad.

BANDA	FRECUENCIA	LONG. DE ONDA
L	1-2 GHz	30-15 cm
S	2-4 GHz	15-7.5 cm
C	4-8 GHz	7.5-3.75 cm
X	8-12.4 GHz	3.75-2.42 cm
Ku	12.4-18 GHz	2.42-1.66 cm
K	18-26.5 GHz	1.66-1.11 cm
Ka	26.5-40 GHz	11.1-7.5 mm
Mm	40-300 GHz	7.5-1 mm

Tabla 4.1.1 Denominación habitual de las bandas de frecuencias en microondas.

A frecuencias superiores nos encontramos con las ondas electromagnéticas correspondientes al infrarrojo, ultravioleta y rayos X (**Tabla 4.1.2**).

BANDA	FRECUENCIA	LONG. DE ONDA	DENOMINACIÓN
	300-800 GHz	1-0.4 mm	Región Submilimetrica
IR	800 GHz-400 THz	0.4 mm-0.8 micras	Infrarrojo
V	400-750 THz	0.8-0.4 micras	Visible
UV	750-10,000 THz	400-12 manómetros	Ultravioleta
		120-0.6 amstrong	Rayos X

Tabla 4.1.2 Denominación de las bandas a frecuencias superiores.

Cada aplicación y cada banda de frecuencias presentan características peculiares que dan origen a unas topologías de antenas muy diversas.

4.1.2 Tipos de antenas.

En una forma amplia y no exhaustiva, los tipos más comunes de antenas se pueden agrupar:

- ***Antenas alambicas***. Se distinguen por estar construidas con hilos conductores que soportan las corrientes que dan origen a los campos radiados. Pueden estar formadas por hilos rectos (dipolo, V, rómbica), espiras (circular, cuadrada o de cualquier forma arbitraria) y hélices. En la **Figura 4.1** se muestran algunos ejemplos de antenas alambicas.

Figura 4.1 Antenas Alambicas.

- ***Antenas de apertura y reflectores***. En ellas la generación de la onda radiada se consigue a partir de una distribución de campos soportada por la antena y se suelen excitar con guías de ondas. Son antenas de apertura las bocinas (piramidales y cónicas), las aperturas y las ranuras sobre planos conductores, y las bocas de guía. El empleo de reflectores, asociado a un alimentador primario, permite disponer de antenas con las prestaciones necesarias para servicios de comunicaciones a grandes distancias, tanto terrestres como espaciales. El reflector más común es el parabólico. En la **Figura 4.1.1** se dan algunos ejemplos de estas antenas.

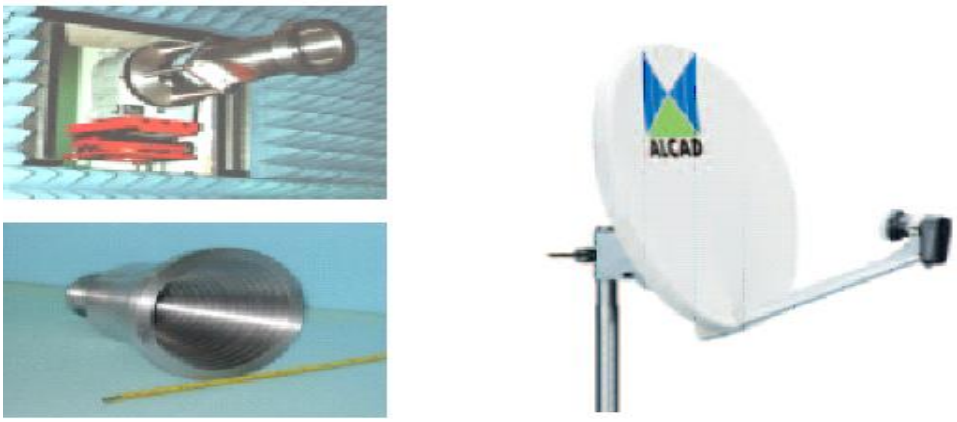


Figura 4.1.1 Antenas de apertura y reflectores.

- ***Agrupaciones de antenas***. En ciertas aplicaciones se requieren características de radiación que pueden lograrse con un solo elemento; sin embargo, con la combinación de varios de ellos se consigue una gran flexibilidad que permite obtenerlas. Estas agrupaciones pueden realizarse combinando, en principio, cualquier antena. En la **Figura 4.1.2** se muestran algunos ejemplos de agrupaciones de antenas.

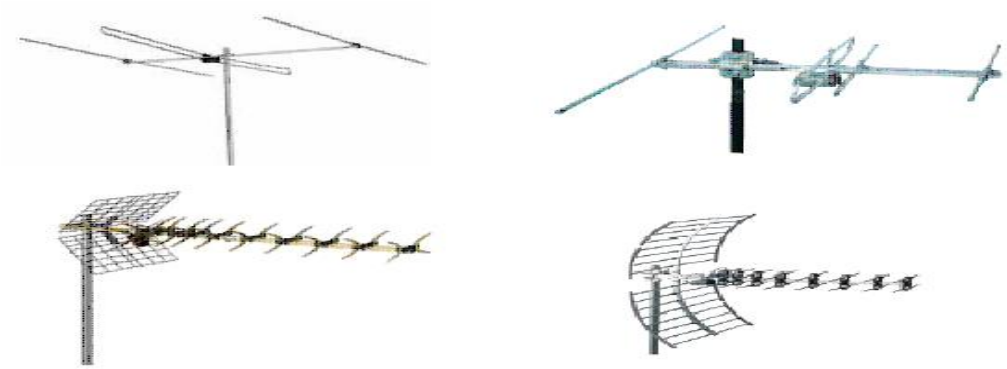


Figura 4.1.2. Agrupaciones de Antenas.

4.2 Parámetros de antenas en transmisión.

Las Antenas forman parte de un sistema más amplio de radiocomunicaciones o radar. Por lo que es importante, caracterizarla con una serie de parámetros que la describan y permitan evaluar el efecto sobre el sistema de una determinada antena, o bien especificar el comportamiento deseado de una antena para incluirla en ese sistema.

4.2.1 Impedancia.

La antena debe conectarse a un transmisor y radiar el máximo de potencia posible con un mínimo de perdidas en ella. La antena y el transmisor deben poseer una buena adaptación de impedancias para una máxima transferencia de potencia en el sentido clásico de circuitos. Habitualmente el transmisor se encuentra alejado de la antena y la conexión se hace mediante una línea de transmisión o guía de ondas, que participa también en esa adaptación, por lo que se considera su impedancia característica, su atenuación y su longitud.
El transmisor produce corrientes y campos que pueden ser medidos en puntos característicos de la antena. A la entrada de la antena puede definirse la *impedancia de entrada* Z_e mediante relaciones tensión-corriente en ese punto. Esta posee una parte real $R_e(\omega)$ y una parte imaginaria $X_e(\omega)$, ambas dependientes en general de la frecuencia. Si Z_e no presenta una parte reactiva a una frecuencia, se dice que es una antena resonante. Dado que la antena radia energía, hay una perdida de potencia hacia el espacio debida a radiación, que puede ser asignada a una resistencia de radiación R_r La resistencia de radiación es la resistencia que, si reemplazara la antena, disiparía exactamente la misma cantidad de potencia de la que irradia la antena.
Todas las perdidas pueden globalizarse en una resistencia de perdidas R_Ω. La resistencia de entrada es la suma de las de radiación y pérdidas.

La impedancia de entrada es un parámetro de gran trascendencia, ya que condiciona las tensiones de los generadores que se deben aplicar para obtener determinados valores de corriente en la antena y, en consecuencia, una determinada potencia radiada. Si la parte reactiva es grande, hay que aplicar tensiones elevadas para obtener corrientes apreciables; si la resistencia de radiación es baja, se requieren elevadas corrientes para tener una potencia radiada importante.

Un ejemplo real puede ser un sistema radiante de radiodifusión de onda media: Para radiar una potencia de 200 kW con una antena de impedancia de entrada 20-j100Ω se requiere una corriente de 100 A y un generador de |V| = 10,200V. Si se compensara la parte reactiva mediante una inductancia, la tensión de generador seria de solo 2,000 V, si bien en ambas reactancias (antena e inductancia) seguirían estando presentes 10,000 V reactivos.

Altos valores de corrientes producen perdidas importantes y elevados valores de tensión pueden producir fugas y descargas entre diversas partes de la antena o con tierra, planteando problemas de forma y aislamiento.

La existencia de perdidas en la antena hace que no toda la potencia entregada por el transmisor sea radiada, por lo que se puede definir un rendimiento o *eficiencia de la antena* η_l, mediante la relación entre la potencia radiada y la entregada, o equivalentemente entre la resistencia de entrada de esa antena, si hubiera sido ideal (sin perdidas), y la que presenta realmente.

$$\eta_l = \frac{p_{radiada}}{p_{entregada}} = \frac{R_r}{R_r + R_\Omega} \qquad \text{(Ec. 4.1)}$$

4.2.2 Intensidad de radiación.

Una de las características fundamentales de una antena es su capacidad para radiar con una cierta direccionalidad, es decir para concentrar la energía radiada en ciertas direcciones del espacio. Será por tanto, conveniente cuantificar este comportamiento con algún parámetro que nos permita establecer una comparación entre distintas antenas. Previamente debemos definir el marco de referencia donde esta situada la antena que queremos caracterizar; para ello emplearemos un sistema de coordenadas que nos permita definir cómodamente una dirección del espacio.

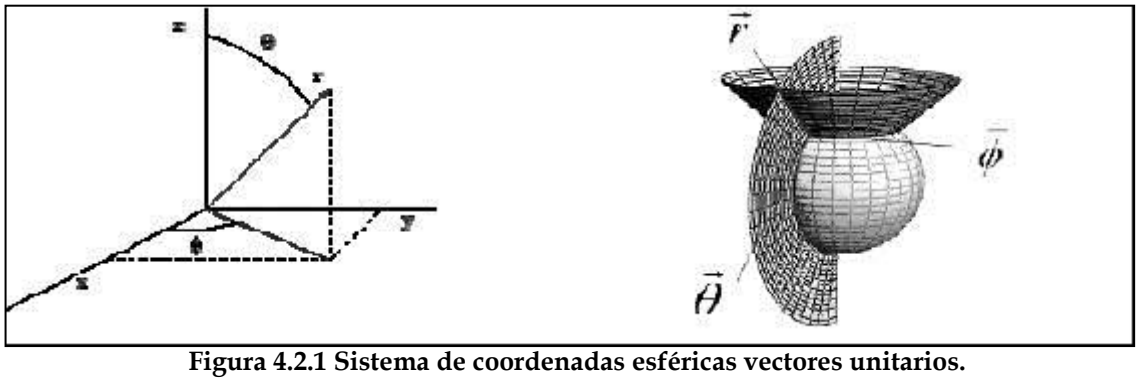


Figura 4.2.1 Sistema de coordenadas esféricas vectores unitarios.

El sistema de coordenadas utilizado habitualmente en antenas es el esférico, ya que mediante la especificación de dos parámetros, los ángulos θ yϕ, queda definido unívocamente una dirección del espacio. En este sistema de coordenadas (**Fig. 4.2.1**) se definen los vectores unitarios $\hat{r}, \hat{\theta}, \hat{\phi}$, que forman una base ortogonal, de forma de que

cualquier vector puede expresarse como una combinación lineal de los tres vectores unitarios.

La onda electromagnética radiada se compone de un campo eléctrico $\vec{E}$ (V/m) y uno magnético $\vec{H}$ (A/m); ambos son magnitudes vectoriales y están ligadas por las ecuaciones de Maxwell.

A partir de los valores eficaces de los campos se obtiene la densidad de flujo por unidad de superficie mediante

$$\vec{\rho}(\theta,\phi) = \mathrm{Re}(\vec{E}\times\vec{H}^{*}) \ \mathrm{W/m^2} \qquad \text{(Ec. 4.2.1)}$$

Donde sé ha supuesto para los campos una variación temporal de la forma $e^{j\omega t}$ y los símbolos *, Re y $\times$ denotan el complejo conjugado, la parte real y el producto vectorial.

Para los campos radiados, los módulos del campo eléctrico y del campo magnético están relacionados por la impedancia característica del medio η, que en el vacío vale $120\pi\ \Omega$.

Por lo general, la densidad de potencia radiada se puede calcular a partir de las componentes transversales del campo eléctrico.

$$\rho(\theta,\phi) = \frac{|E_\theta|^2 + |E_\phi|^2}{\eta} \qquad \text{(Ec.4.2.2)}$$

La potencia total radiada se puede obtener como la integral de la densidad de potencia en una superficie esférica que encierra a la antena.

$$P_r = \iint_s \vec{\rho}(\theta,\phi).d\vec{s} \qquad \text{(Ec. 4.2.3)}$$

La intensidad de radiación es la potencia radiada por unidad de ángulo sólido en una determinada dirección; sus unidades son vatios por estereorradián y a grandes distancias tiene la propiedad de ser independiente de la distancia a la que se encuentre la antena.

La relación entre la intensidad de radiación y la densidad de potencia radiada es

$$K(\theta,\phi) = \rho(\theta,\phi)r^2 \qquad \text{(Ec. 4.2.4)}$$

Y la potencia total radiada también se puede calcular integrando la intensidad de radiación (Ec. 4.2.4) en todas las direcciones el espacio

$$P_r = \iint_{4\pi} K(\theta,\phi).d\Omega \qquad \text{(Ec. 4.2.5)}$$

Al ser el diferencial de ángulo en coordenadas esféricas

$$d\Omega = ds/r^2 = sen\theta.d\theta d\phi \qquad \text{(Ec. 4.2.6)}$$

4.2.3 Diagrama de radiación.

Un diagrama es una representación grafica de las propiedades de radiación de la antena en función de las distintas direcciones del espacio, a una distancia fija. Normalmente se utilizara un sistema de coordenadas esféricas. Con la antena situada en el origen y

manteniendo constante la distancia se expresara el campo eléctrico en función de las variables angulares (θ,ϕ). Como el campo es una magnitud vectorial, habrá que determinar en cada punto de la esfera de radio constante el valor de dos componentes ortogonales, habitualmente según θ,ϕ.

El campo magnético se deriva del eléctrico, la representación podría realizarse a partir de cualquiera de las dos, siendo una norma habitual que los diagramas se refieran al campo eléctrico.

La densidad de potencia es proporcional al cuadrado del modulo del campo eléctrico, por lo que la represtación grafica de un diagrama de potencia contiene la misma información que un diagrama de radiación de campo.

En determinadas circunstancias puede ser necesaria la representación grafica de la fase de E (θ, ϕ), además de la amplitud de las dos componentes. Dicha representación se denomina el diagrama de fase de la antena.

Al observar a gran distancia una antena, se vería su radiación como si proviniera de un punto, es decir, los frentes de onda seria esféricos. A ese punto, centro de curvatura de las superficies de fase constante, se le denomina el centro de fase de la antena.

El diagrama de radiación se puede representar en forma tridimensional utilizando técnicas graficas diversas según la **Fig. 4.2.2**.

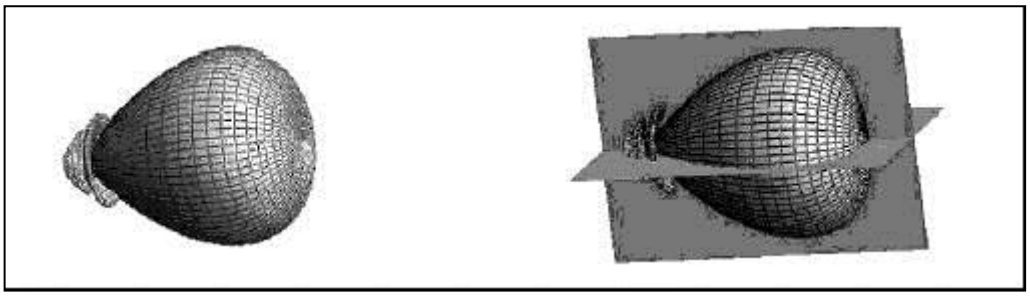


Figura 4.2.2 Representación tridimensional.

Si bien la información de la radiación es tridimensional, puede ser de interés, y en muchos casos suficiente, representar un corte del diagrama. Los cortes más habituales son los que siguen los meridianos en una hipotética esfera (cortes para ϕ constante) o paralelos (cortes con θ constante).

Para antenas linealmente polarizadas se define el plano **E** como el que forman la dirección de máxima radiación, y el campo eléctrico en dicha dirección. Análogamente, el plano **H** es el formado por la dirección de máxima radiación y el campo magnético en dicha dirección. Ambos planos son perpendiculares y su intersección determina una línea que define la dirección de máxima radiación de la antena.

Los cortes bidimensionales del diagrama de radiación se pueden representar en coordenadas polares o cartesianas (**Fig. 4.2.3**). La representación en coordenadas cartesianas permite observar detalles en antenas muy directivas, mientras el diagrama polar suministra una información más clara de la distribución de la potencia en las diferentes direcciones del espacio.

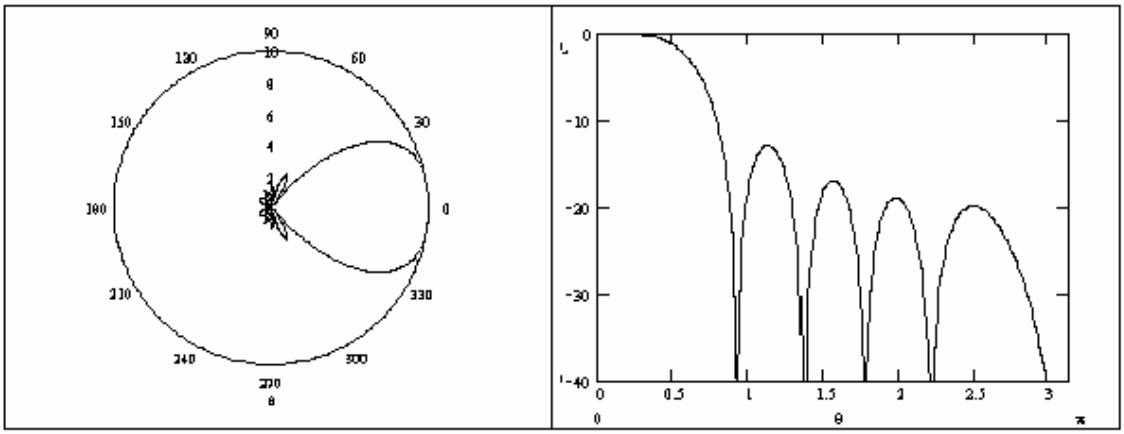


Figura 4.2.3 Diagrama de Radiación en Coordenadas Polares y Rectangulares

4.2.4 Antena Isotropica.

Se denomina antena isótropa a una antena ideal que radie la misma intensidad de radiación en todas las direcciones del espacio (**Fig. 4.2.4**). Aunque no existe una antena de estas características es de gran utilidad para definir los parámetros de la siguiente sección.

Figura 4.2.4 Antena Isotropica.

4.2.5 Antena Omnidireccional.

Si un diagrama de radiación representa simetría de revolución entorno a un eje se dice que la antena es omnidireccional (**Figura 4.2.5**).

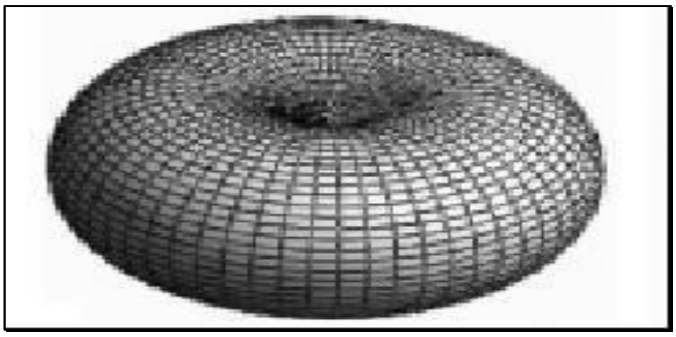


Figura 4.2.5 Antena Omnidireccional.

4.2.6 Directividad.

La directividad D de una antena se define como la relación entre la densidad de potencia radiada en una dirección, a una distancia dada, y la densidad de potencia que radiaría a esa misma distancia una antena isótropa que radiase la misma potencia que la antena.

$$D(\theta,\phi) = \frac{\rho(\theta,\phi)}{P_r/(4\pi r^2)} \qquad \text{(Ec.4.2.7)}$$

Si no especifica la dirección angular, se sobreentiende que la directividad se refiere a la dirección de máxima radiación.

$$D(\theta,\phi) = \frac{\rho_{\max}}{P_r/(4\pi r^2)} \qquad \text{(Ec. 4.2.8)}$$

Un segundo termino directamente relacionado con la directividad es la *ganancia* de la antena G. Su definición es semejante, pero la comparación no se establece con la potencia radiada, sino con la potencia entregada a la antena. Ello permite tener en cuenta posibles perdidas en la antena, ya que entonces no toda la potencia entregada es radiada al espacio. La ganancia y la directividad están relacionadas, en consecuencia, por la eficiencia de la antena.

$$G(\theta,\phi) = \eta_l DG(\theta,\phi) \qquad \text{(Ec. 4.2.9)}$$

Si la antena no posee pérdidas, cosa habitual a altas frecuencias, ambos parámetros son equivalentes.

4.2.7 Polarización.

La polarización de una onda es la figura geométrica determinada por el extremo del vector que representa al vector de campo eléctrico radiado y su sentido de giro, visto por un observador situado en la antena y en un plano estacionario perpendicular a la dirección de propagación, mientras la onda circule a través del plano. Recuérdese que el vector de campo eléctrico es perpendicular a la dirección de desplazamiento y al vector de campo magnético. Para ondas con variación sinusoidal dicha figura es en general una elipse. Hay una serie de casos particulares. Si la figura trazada es una recta, la onda se denomina linealmente polarizada, si es un círculo se denomina circularmente polarizada.

El sentido de giro del campo eléctrico, para una onda que se aleja del observador, determina si la onda está polarizada circularmente a derecha o a izquierda. Si el sentido de giro coincide con las agujas del reloj, la polarización es circular a la derecha. Si el sentido de giro es contrario a las agujas del reloj, la polarización es circular a la izquierda. El mismo convenio aplica a las ondas con polarización elíptica.
Una onda radiada por una antena centrada en el eje z y en un punto cualquiera se puede poner como contribución de dos componentes ortogonales entre sí, donde también los campos se han representado en notación fasorial. Por otra parte estas componentes ortogonales pueden ser lineales o circulares.

En la Figura 4.2.6. Se representa un ejemplo en el caso de que la observación se realice en el eje z.

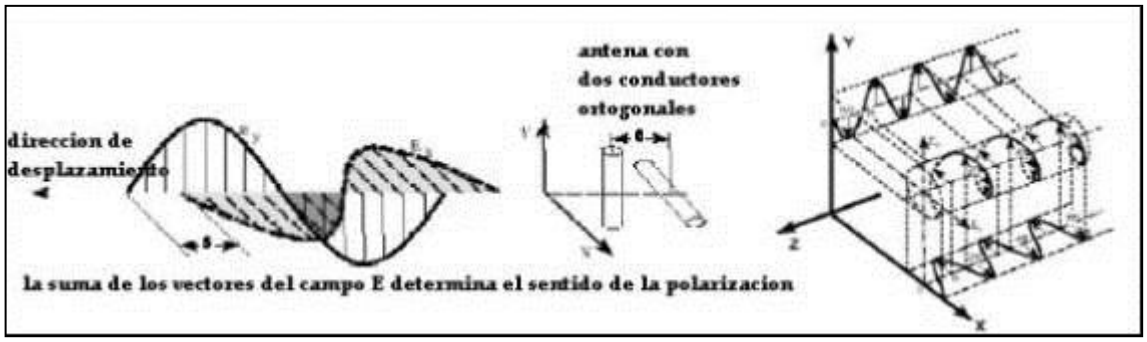


Figura 4.2.6 Representación del campo eléctrico como composición de dos campos ortogonales, en el caso de observación en el eje z.

$$\vec{E}(\theta,\phi) = E_\theta(\theta,\phi)\cdot\vec{\theta} + E_\phi(\theta,\phi)\cdot\vec{\phi} \quad \begin{cases} E_\theta(\theta,\phi) = |E_\theta| e^{j\delta_\theta} \\ E_\phi(\theta,\phi) = |E_\phi| e^{j\delta_\phi} \end{cases} \qquad \text{(Ec. 4.2.10)}$$

Para determinar la variación temporal es suficiente con determinar el valor real instantáneo de cada una de las componentes.

(Ec. 4.2.11)

$$E_{\theta i} = |E_\theta| \cos(\omega t + \delta_\theta)$$
$$E_{\phi i} = |E_\phi| \cos(\omega t + \delta_\phi)$$

Si en la ecuación anterior se elimina la variable tiempo, se obtiene la siguiente relación:

$$\left(\frac{E_{\theta i}}{|E_\theta|}\right)^2 - 2\frac{E_{\theta i}}{|E_\theta|}\frac{E_{\phi i}}{|E_\phi|}\cos\delta + \left(\frac{E_{\phi i}}{|E_\phi|}\right)^2 = sen^2\delta \qquad \text{(Ec.4.2.12)}$$

En donde $\delta = \delta\varphi - \delta\theta$. Si se representan ambas componentes del campo eléctrico en cada instante de tiempo es, en general, una elipse. Bajo ciertas condiciones la elipse puede plegarse en una línea recta, en este caso la polarización se denomina lineal.

En esta **Figura 4.2.7** se representa al vector de campo eléctrico E en distintas relaciones de amplitud relativa y ángulo de fase de sus componentes Ey y Ex, visto desde un observador que "mira" hacia la antena emisora.

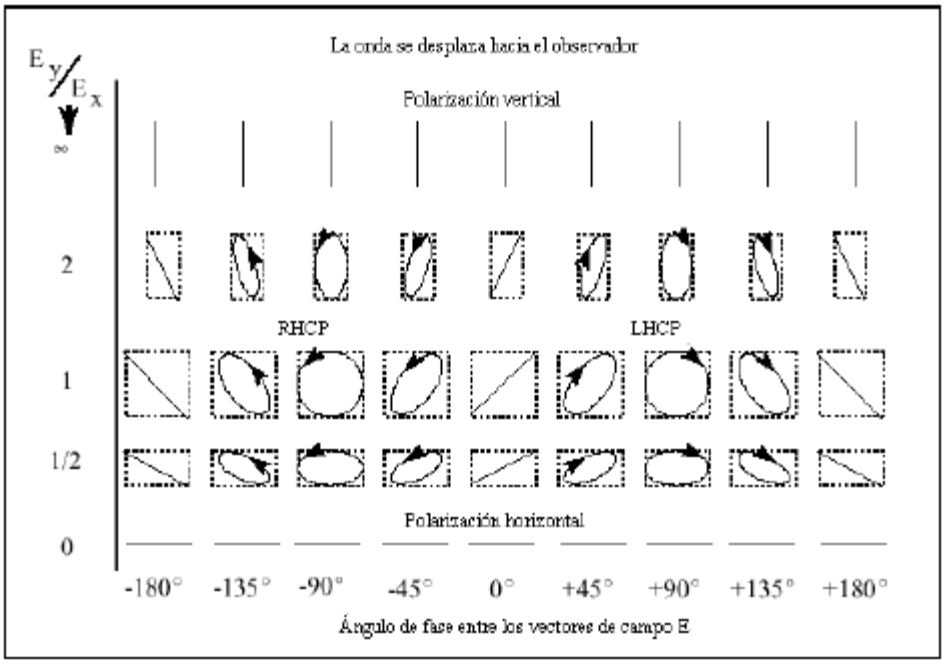


Figura 4.2.7 Componentes del Campo Eléctrico.

En el otro extremo, cuando las dos componentes tienen la misma magnitud y desfase de 90º, la elipse es circular, tal como se ha mostrado. De manera que las polarizaciones lineal y circular son dos casos especiales de la polarización elíptica. La polarización lineal puede clasificarse adicionalmente en vertical, horizontal o inclinada.
Cualquier onda se puede descomponer en dos componentes ortogonales entre sí, estas componentes pueden ser polarizaciones lineales ortogonales, sin más que proyectar el campo eléctrico sobre vectores unitarios orientados según dichas direcciones. Aplicando el mismo principio, cualquier onda se puede descomponer en dos ondas polarizadas circularmente hacia la derecha o hacia la izquierda. Para una antena de polarización lineal, el patrón de radiación se compone de repuestas tanto copolarizada como contra polarizada (no deseada). La calidad de la polarización se expresa por la razón entre estas dos respuestas, debe ser típicamente grande (30 dB o mayor) para aplicaciones como la interferencia (perturbación) contrapolarizada. Para las aplicaciones generales, esta razón indica pérdida de potencia debido a una mala polarización. Para las antenas con polarización circular, los patrones de radiación se cogen a partir de una antena de referencia linealmente polarizada.

Se define la relación axial de una onda polarizada elípticamente, como la relación entre los ejes mayor y menor de la elipse de polarización. La relación axial toma valores comprendidos entre 1 e infinito.
El sentido de giro de Polarización, tanto en la polarización circular como en la elíptica, puede ser hacia la derecha (CW, RHCP) para $\delta < 0$, o hacia la izquierda (CCW, LHCP) para $\delta > 0$.
La condición para que la polarización sea lineal es que $\delta = 0º$, o que $\delta = 180º$, o bien que $E\varphi = 0$, o que $E\theta = 0$. Para que sea circular se debe cumplir que $|E\varphi| = |E\theta|$ y que $\delta = +90º$ ó $\delta = -90$

4.2.8 Ancho de banda.

Todas las antenas, debido a su geometría finita, están limitadas a operar satisfactoriamente en una banda o margen de frecuencias. Este intervalo de frecuencias, en el que un parámetro de antena determinado no sobrepase los límites prefijados, se conoce como el ancho de banda de la antena. Puede ser definido respecto a múltiples parámetros (diagrama de radiación, directividad, impedancia, etc.). El ancho de banda de la antena lo impondrá el sistema del que forme parte y afectara al parámetro más sensible o crítico en la aplicación. Para su especificación los parámetros pueden dividirse en dos grupos, según su relación con el diagrama o con la impedancia.
En el primero de ellos tenemos la directividad, la pureza de Polarizacion, el ancho de haz, el nivel de lóbulo principal a secundario y la dirección de máxima radiación.
En el segundo la impedancia de entrada, el coeficiente de reflexión y la relación de onda estacionaria.

4.3 Parámetros de antenas en recepción

Una antena capta de una onda incidente sobre ella parte de la potencia que transporta y la transfiere al receptor. La antena actúa como un sensor e interacciona con la onda y con el receptor, dando origen a una familia de parámetros asociados con la conexión de los circuitos y otra vinculada a la interacción electromagnética con la onda incidente.

4.3.1 Adaptación

Desde los terminales de la antena, el receptor se ve como una impedancia de carga $Z_L=R_L+jX_L$, mientras que el receptor ve a la antena como un generador ideal de tensión V_{CA} e impedancia $Z_a = R_a + jX_a$. La transferencia de potencia será máxima cuando haya adaptación conjugada ($Z_L = Z_a$). Entonces la potencia entregada por la antena a la carga será

$$P_{L\max} = \frac{|V_{CA}|^2}{4R_a} \qquad \text{(Ec. 4.3.1)}$$

En general, si no hay desacople tendremos:

$$P_L = P_{Lmax}C_a \qquad \text{(Ec. 4.3.2)}$$

Donde C_a es el coeficiente de desacople de impedancias, dado por:

$$C_a = \frac{4R_aR_L}{(R_a+R_L)^2+(X_a+X_L)^2} \qquad \text{(Ec. 4.3.3)}$$

Normalmente, entre la antena y el receptor existe una línea de transmisión de impedancia característica Z_0. En este caso, el coeficiente de desacople vale también $1 - |\rho_L|^2$, donde ρ_L es el coeficiente de reflexión.

4.3.2 Área y longitud efectiva

La antena extrae potencia del frente de onda incidente, por lo que presenta una cierta área de captación o área efectiva A_{ef}, definida como la relación entre la potencia que entrega la antena a su carga (supuesta por esta definición sin perdidas y adaptada a la carga) y la densidad de potencia incidente que representa físicamente la porción del frente de onda que la antena ha de interceptar y drenar de el toda la potencia contenida hacia la carga.

$$A_{ef} = \frac{P_L}{\rho} \qquad \text{(Ec. 4.3.4)}$$

La definición anterior lleva implícita la dependencia del área efectiva con la impedancia de carga, acople de impedancias y la polarización de onda, si sustituimos (Ec. 4.3.1) en (Ec. 4.3.4) y tenemos en cuenta que $\rho = |E|^2 / \eta$, resulta:

$$A_{ef} = \frac{|V_{CA}|^2}{4R_r \rho} = \frac{|V_{CA}|^2 \eta}{|E|^2 4R_r} = \frac{\ell_{ef}{}^2 \eta}{4R_r} \qquad \text{(Ec. 4.3.5)}$$

Donde se ha introducido un nuevo parámetro, la longitud efectiva ℓ_{ef}, mediante la relación entre la tensión inducida en circuito abierto en bornes de la antena y la intensidad del campo incidente en la onda

$$\ell_{ef} = \frac{|V_{ca}|}{|E|} \qquad \text{(Ec. 4.3.6)}$$

Esta definición lleva implícita una dependencia con la polarización de la onda. La longitud y el área efectiva están definidas a partir de magnitudes eléctricas y no coinciden necesariamente con las dimensiones reales de las antenas, si bien en algunos tipos de ellas se guardan una relación directa.

5
TIPOS DE ANTENAS.

5.1 ANTENA DIPOLO

5.1.1 Dipolo Elemental.

Es un radiador elemental, cuya longitud es tan inferior a la longitud de onda, que se considera a la corriente como uniforme en toda la longitud del dipolo elemental.

Supongamos que el dipolo coincide con el eje Oz, y que su centro coincide con el centro de una esfera que se encuentra en una zona distante y tiene un radio r (Fig. 5.1). El plano que pasa por el centro de la esfera perpendicularmente al eje Oz se llama ecuatorial y los planos que pasan por el eje Oz se llaman meridionales.

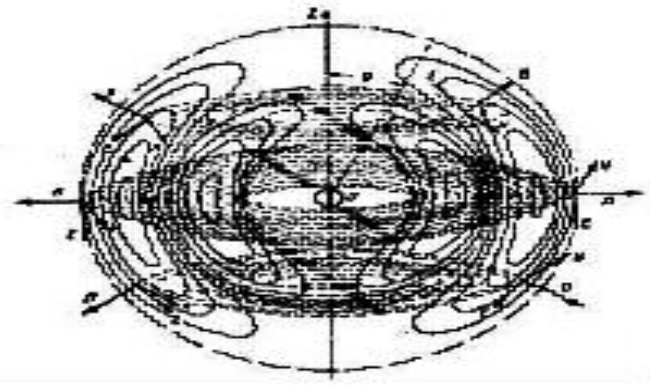


Figura 5.1 Campos de Radiación del Dipolo Elemental.

Como puede verse en la figura 5.1 las líneas de fuerza del campo eléctrico del dipolo se encuentran en el plano meridional. Las líneas del campo magnético, que tienen la forma de circunferencias con centro en el eje Oz, están dispuestas en planos perpendiculares al eje Oz. Por esta razón, al plano meridional se le llama plano E, y al ecuatorial, plano H. De acuerdo con la posición del vector E (en el plano meridional que pasa por un punto dado en el espacio) y del vector H (en el mismo punto, perpendicularmente al plano meridional) el vector de Poynting, indica que las ondas electromagnéticas se propagan en la dirección de los radios que parten del centro de la esfera. En otras palabras, en el caso dado se obtiene una onda esférica que, a suficiente distancia del dipolo, se puede considerar como plana.

Esta onda es progresiva. A la distancia **r**, que corresponde a la zona lejana, y con una longitud de onda λ, los valores instantáneos de las tensiones de los campos eléctricos y magnéticos se expresan, respectivamente, con las formulas[1]:

$$E_i = E_m sen(\varpi t - \beta r) = E_m sen\left(\varpi t - \frac{2\pi}{\lambda} r\right) \quad \text{(Ec. 5.1.1)}$$

$$H_i = H_m sen(\varpi t - \beta r) = H_m sen\left(\varpi t - \frac{2\pi}{\lambda} r\right) \quad \text{(Ec. 5.1.2)}$$

5.1.2 Ecuación del dipolo elemental

El análisis matemático del apéndice A demuestra que en un dipolo elemental de longitud l con una amplitud de corriente I_m, la amplitud de la tensión del campo eléctrico será igual a[3]:

$$E_m = \frac{60\pi I d\ell sen\,\theta}{\lambda r} \quad \text{(Ec. A.13)}$$

Y la del campo magnético

$$H_m = \frac{I d\ell sen\,\theta}{2\lambda r} \quad \text{(Ec. A.14)}$$

En la que θ es el ángulo cenital formado por el eje del dipolo y el radio de la esfera que pasa por un punto dado del espacio.
Basándose en las formulas citadas se puede asegurar que las ondas electromagnéticas emitidas por el dipolo elemental tiene las siguientes características:

1. Las amplitudes de las intensidades de campo del dipolo son directamente proporcionales a la amplitud de la corriente que hay en el, eso se explica por el hecho de que la corriente del dipolo determina la potencia suministrada, y en consecuencia, también la potencia de radiación.

2. Las intensidades de campo son directamente proporcionales a la relación l/λ, lo que corresponde a la conocida propiedad de amplificar la radiación de ondas electromagnéticas aumentando la longitud l del radiador con respecto a la longitud de onda λ (lo que es cierto dentro de determinados limites).

3. Las intensidades de campo son inversamente proporcionales a la distancia r al dipolo; esta propiedad es característica, de la onda esférica.

4. A medida que se disminuye el ángulo cenital θ desde 90˚ hasta 0, las intensidades de campo disminuyen, según la ley sen θ, desde un máximo hasta cero. Podemos convencernos de esta no uniformidad de radiación comparando en la figura 5.1 la densidad de las líneas de fuerza en las diferentes direcciones radiales.

5.1.3 Diagrama de radiación del dipolo elemental.

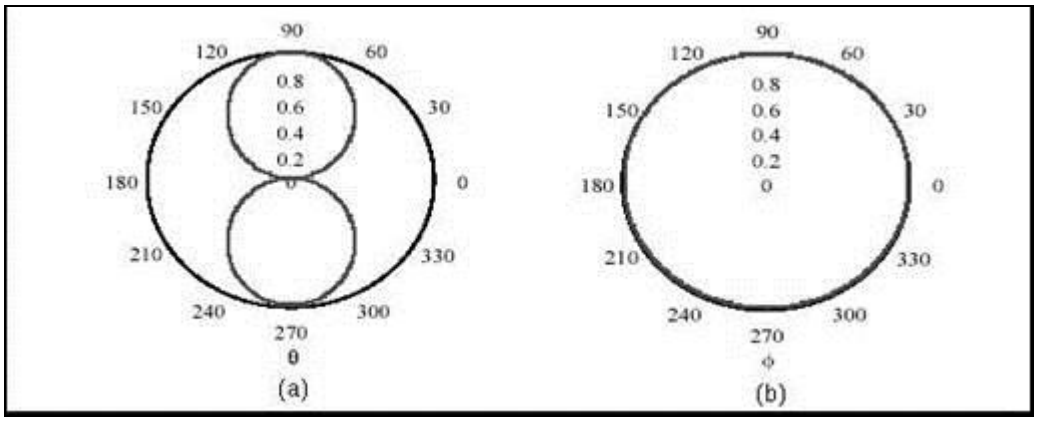


Figura 5.2 Diagramas de Radiación del Dipolo Elemental en los Sistemas de Coordenadas Polar (a), y Rectangular (b).

En el diagrama de radiación del dipolo elemental el plano meridional (plano E), el diagrama tiene forma de ocho Fig. 5.2 (a) mientras que en el plano ecuatorial (plano H) la tiene de circunferencia Fig. 5.2 (b). Reuniendo los diagramas polares en los planos E y H obtendremos el diagrama espacial de radiación del dipolo elemental en forma de un toroide Fig.5.3.

.

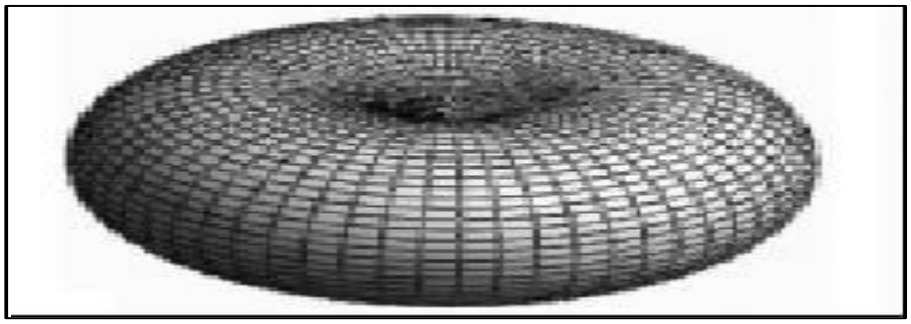


Figura 5.3 Diagrama de Radiación Volumétrico del Dipolo Elemental.

5.1.4 Resistencia de Radiación del Dipolo Elemental.

Determinemos la resistencia de radiación R_r, la potencia total radiada por el elemento de corriente puede valorarse integrando el vector radial de Poynting en una superficie esférica centrada en el elemento. La potencia es independiente del ángulo azimutal ϕ, de manera que como elemento de área de la capa esférica se tomara un diferencial de área según la figura 5.4, en la que $da = 2\pi r^2 \text{sen}\,\theta$, entonces la potencia radial total será[4]

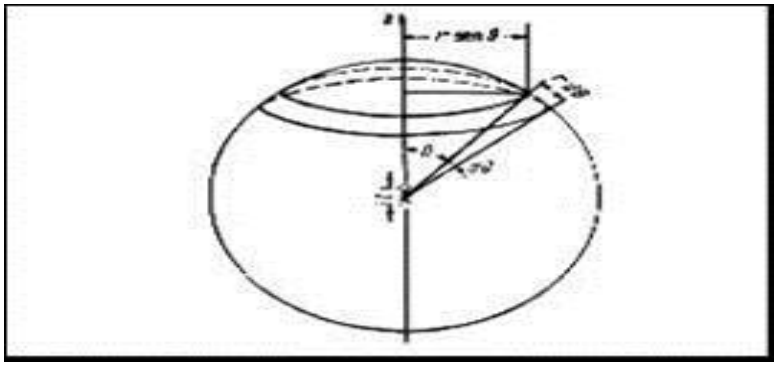


Figura 5.4 El dipolo elemental rodeado de una esfera de radio r.

$$Potencia = \oint_{superficie} P_{r(av)} da = \int_0^{\pi} \frac{\eta}{2}\left(\frac{\varpi I dl sen\theta}{4\pi r c}\right)^2 2\pi r^2 sen\theta d\theta \qquad \text{(Ec. 5.1.3)}$$

Donde $P_{r(av)}$ se determino en el apéndice A

$$\Pr(av) = \frac{\eta \varpi^2 I^2 dl^2}{16\pi c^2}\int_0^{\pi} sen^3\theta d\theta \qquad \text{(Ec. A-12)}$$

$$= \frac{\eta \varpi^2 I^2 dl^2}{16\pi c^2}\left[-\frac{\cos\theta}{3}(sen^2\theta + 2)\right]_0^{\pi}$$

$$= \frac{\eta \varpi^2 I^2 dl^2}{12\pi c^2}$$

En esta expresión I es la corriente máxima. La potencia radiada en función de la corriente eficaz será

$$Potencia = \frac{\eta \varpi^2 I_{ef}^2 dl^2}{6\pi c^2}$$

$$Potencia = 80\pi^2\left(\frac{dl}{\lambda}\right)^2 I_{ef}^2 \qquad \text{(Ec. 5.1.4)}$$

El coeficiente de I_{ef}^2 tiene las dimensiones de una resistencia y se denomina resistencia de radiación del elemento de corriente. Así para un elemento de corriente

$$R_r = 80\pi^2\left(\frac{dl}{\lambda}\right)^2 \qquad \text{(Ec. 5.1.5)}$$

5.1.5 Dipolos Simétricos.

El dipolo simétrico se diferencia del elemental por la distribución no uniforme de la corriente a lo largo de su longitud. No obstante, la teoría el dipolo elemental permite descubrir las propiedades del dipolo simétrico, pues este último puede ser representado por un número de dipolos elementales infinitamente grande. En cualquier punto del espacio se interfieren los campos de los dipolos elementales, y según las diferencias de

fases entre ellos, el campo resultante de los componentes de los elementos separados del dipolo simétrico se amplifica o se debilita.

Para calcular los campos electromagnéticos de antenas mas largas es necesario conocer la distribución de la corriente a lo largo de las mismas. Esta información se obtendría resolviendo las ecuaciones de Maxwell sujetas a las condiciones aproximadas de los límites a lo largo de la antena. Desconociendo la corriente de la antena es posible suponer una cierta distribución, y de estas calcular las distribuciones aproximadas de los campos.

Esto determina el método de la investigación al cual someteremos en el examen de casi todos los radiadores: primero se divide el radiador en otros elementales cuyos campos individuales son conocidos (mayor detalle en Apéndice A); esos campos se suman tomando en consideración como están distribuidas, en amplitud y fase, las corrientes en el radiador, y cual es la diferencia de camino que toman los rayos desde los elementos del dipolo hasta el punto dado del espacio; en la expresión del campo resultante que obtendremos se pone de relieve la amplitud y el factor que determina la dependencia de dicha amplitud con respecto a la dirección, es decir la función de directividad; esta ultima se normaliza (si es necesario) y se representa en forma de diagrama. Como puede verse, para investigar un dipolo simétrico es preciso, ante todo, conocer la ley de distribución de la corriente en el. En la práctica se resuelve este problema partiendo de la analogía entre el dipolo y una línea bifilar. Esta analogía ocasiona algunos errores[1]:

1. Una línea bifilar normal es un circuito con constantes uniformemente distribuidas, mientras que la antena tiene una desigualdad de distribución de parámetros. En particular, en el dipolo simétrico varia la distancia entre las secciones simétricas del conductor y sus parámetros lineales a medida que nos vamos alejando de los terminales del generador.

2. El campo eléctrico de una línea bifilar es un campo de potencial, en tanto que el de la antena es rotacional. Como la energía consumida en el traslado de una carga en el campo rotacional de un oscilador depende de la trayectoria de la carga, la diferencia de potencial entre dos puntos del dipolo queda indeterminada. En un dipolo solo puede utilizarse el concepto de tensión cuando la longitud del dipolo es pequeña en comparación con la longitud de onda, o cuando se trata de la diferencia de potencial en los bornes de la antena, donde el campo eléctrico es cercano al campo de potencial. En los demás casos hablaremos de los potenciales de la antena como de una magnitud proporcional a la densidad superficial de las cargas.

3. la línea no irradia campo electromagnético, en tanto que el dipolo lo irradia, y como este campo influye a su vez sobre la corriente del dipolo, la distribución de la corriente es algo diferente que en una línea.

Los errores que acarrea el uso de la mencionada analogía son mayores cuanto la mayor parte de longitud de onda alcance las medidas transversales del dipolo. La experiencia demuestra que en las bandas de onda larga, media y en muchos casos cortas, los errores no traspasan los límites tolerables.

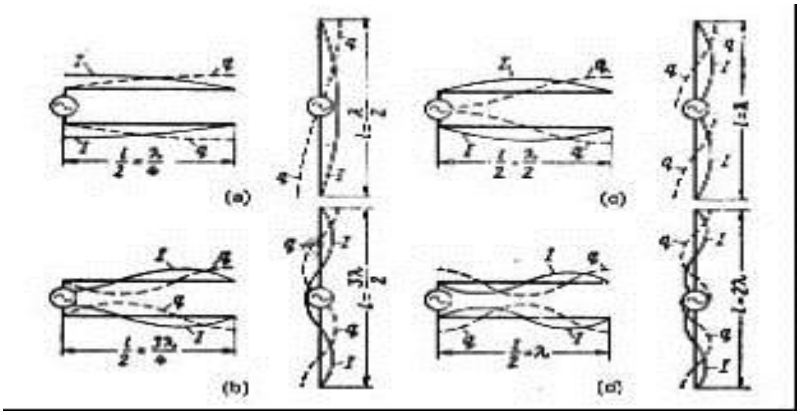


Figura 5.5 Distribución de los valores máximos de la corriente y de la carga a lo largo de las lineas abiertas .

En la figura 5.5 se muestra la distribución de la carga y la amplitud de la corriente en una línea abierta y en el dipolo que corresponde a la misma por su longitud, para $l = \lambda/2$, λ, $3\lambda/2$, 2λ. En los extremos de la línea, la corriente es igual a cero y las cargas son máximas (nodo de corriente y antinodo de carga). A medida que nos aproximamos a los terminales del generador, la corriente y las cargas de la línea varían según las leyes de las ondas estacionarias.

El paso de la línea al dipolo se realiza con giro de 90˚ de los conductores hacia lados opuestos, conservándose las curvas de variación de la corriente y de la carga. Como resultado, se pone de manifiesto que en cortes simétricos de un dipolo de cualquier longitud, las cargas son numéricamente iguales y de signo opuesto, mientras que las corrientes son iguales en magnitud y coinciden en fase. Esta última circunstancia es tan importante que conviene recordar: un dipolo simétrico es el conductor rectilíneo en cuyos puntos simétricos las corrientes son iguales y coinciden en dirección y fase.

Si a lo largo de un dipolo se dispone un número entero de semiondas, el dipolo recibe el nombre de antena armónica.

5.1.6 Ecuación del dipolo simétrico.

Se supondrá que la corriente esta distribuida sinusoidalmente, como se indica en la figura 5.6. Entones

$$I = I_m sen\beta(h - z) \qquad z>0 \qquad \text{(Ec. 5.1.6)}$$

$$I = I_m sen\beta(h + z) \qquad z<0 \qquad \text{(Ec. 5.1.7)}$$

En donde I_m es el valor de la corriente en el vientre o corriente máxima. La expresión del potencial vector en un punto P debido al elemento de corriente Idz será:

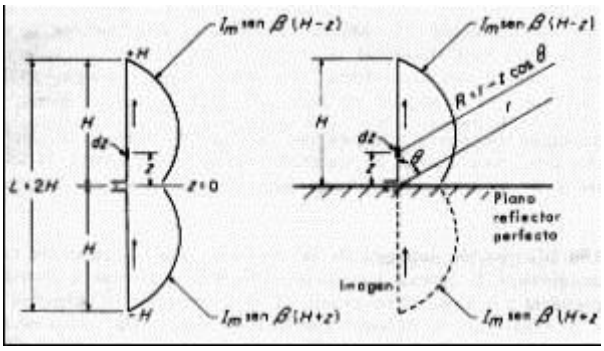


Figura 5.6 Dipolo con alimentación central y una distribución supuesta de corriente sinusoidal.

$$dA_z = \frac{\mu I e^{-j\beta R} dz}{4\pi R} \qquad \text{(Ec. 5.1.8)}$$

Donde R es la distancia desde el elemento de corriente al punto P. El potencial vector total debido a todos los elementos de corriente será

$$A_z = \frac{\mu}{4\pi}\int_{-H}^{0} \frac{I_m sen\beta(h+z)e^{-j\beta R}dz}{R} + \frac{\mu}{4\pi}\int_{0}^{H} \frac{I_m sen\beta(h-z)e^{-j\beta R}dz}{R} \qquad \text{(Ec. 5.1.9)}$$

La ecuación 5.1.9, de acuerdo con el apéndice B, se reduce a la forma

$$A_z = \frac{\mu I_m e^{-j\beta r}}{2\pi\beta r}\left[\frac{\cos(\beta H\cos\theta) - \cos\beta H}{sen^2\theta}\right] \qquad \text{(Ec. B.3)}$$

Lo mismo para el campo eléctrico del dipolo simétrico, según en el **apéndice B (Ec. B.6)** e incluyendo el factor tiempo así en forma general:

$$E_\theta = \frac{j60 I_m e^{-j\beta r}}{r}\left[\frac{\cos\left(\frac{2\pi H}{\lambda}\cos\theta\right) - \cos\frac{2\pi H}{\lambda}}{sen\theta}\right] sen(\varpi t - \beta r) \qquad \text{(Ec. 5.1.10)}$$

El factor sen (ωt - βr) indica que el dipolo simétrico emite ondas progresivas. En este factor, el ángulo de fase ωt-βr depende de la distancia r y no depende de las coordenadas angulares. Lo primero significa que el punto medio O es el punto equivalente de radiación (el centro de fase de todo el dipolo), y lo segundo significa que las ondas radiadas son esféricas.

5.1.7 Diagrama de radiación del dipolo simétrico

De la ecuación 5.1.10 tomaremos como función de directividad del dipolo

$$f_\theta = \frac{\cos\left(\frac{2\pi H}{\lambda}\cos\theta\right) - \cos\frac{2\pi H}{\lambda}}{sen\,\theta} \qquad \text{(Ec. 5.1.11)}$$

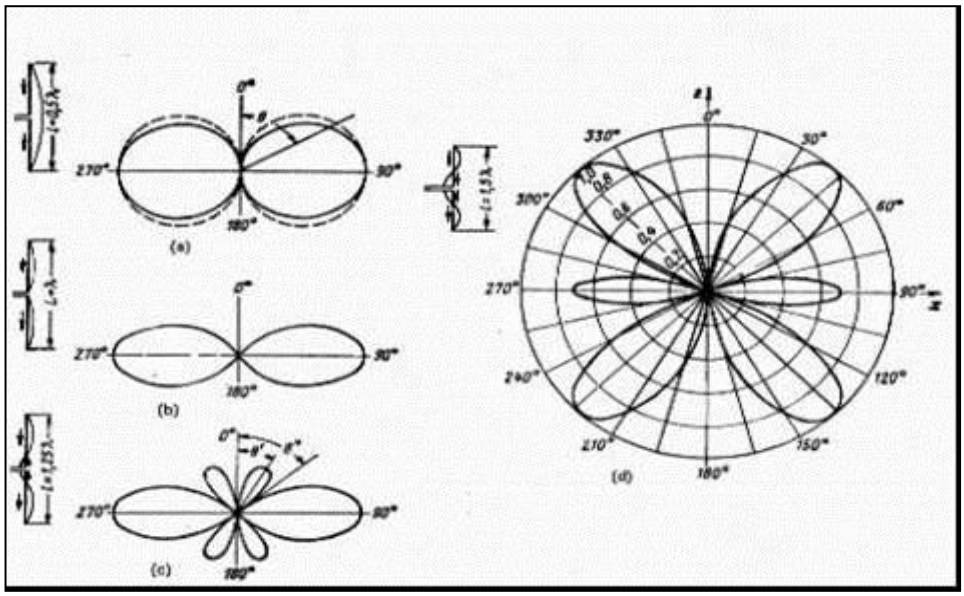


Figura 5.7 Diagramas de directividad, en el plano meridional, de un dipolo simétrico de longitud: a) l = 0.5λ, b) l = λ, c) l = 1.25 λ, d) l = 1.5 λ.

La figura 5.7 representa el diagrama de radiación normalizado de un dipolo simétrico. La ecuación de estos diagramas, F(θ), se ha compuesto dividiendo la función f(θ) por su máximo $F(\theta)= f(\theta)/f_m(\theta)$.

En el dipolo de media onda (Fig.5.7a) las funciones de directividad normalizada y no normalizada son iguales, el diagrama de esta función se diferencia muy poco del correspondiente diagrama del dipolo elemental, y por ello en los dipolos simétricos cuya longitud no supere λ/2, puede utilizarse una expresión mas sencilla: F(θ) = sen θ.

En el dipolo de longitud l = λ (Fig. 5.7b), en este dipolo se observa un considerable aumento de directividad en comparación con el de l = λ/2, lo que se puede explicar con el doble aumento del numero de dipolos elementales, cada uno de los cuales posee algunas propiedades direccionales en el plano meridional.

Con una longitud de dipolo l = 1.25λ (Fig.5.7c) aparece una sección (una quinta parte) donde la corriente tiene sentido contrario con respecto a la parte principal del dipolo. Debido a esto, el campo resultante del dipolo disminuye más rápidamente, a medida que nos apartamos del plano ecuatorial.

Al extender el dipolo hasta l = 1.5 λ (Fig.5.7d), la sección con corriente en sentido opuesto aumenta hasta 0.5 λ, a consecuencia de lo cual la radiación en el plano ecuatorial diminuye aun mas, mientras el lóbulo complementario del diagrama de directividad aumenta de tamaño acercándose en su máximo al eje del dipolo.

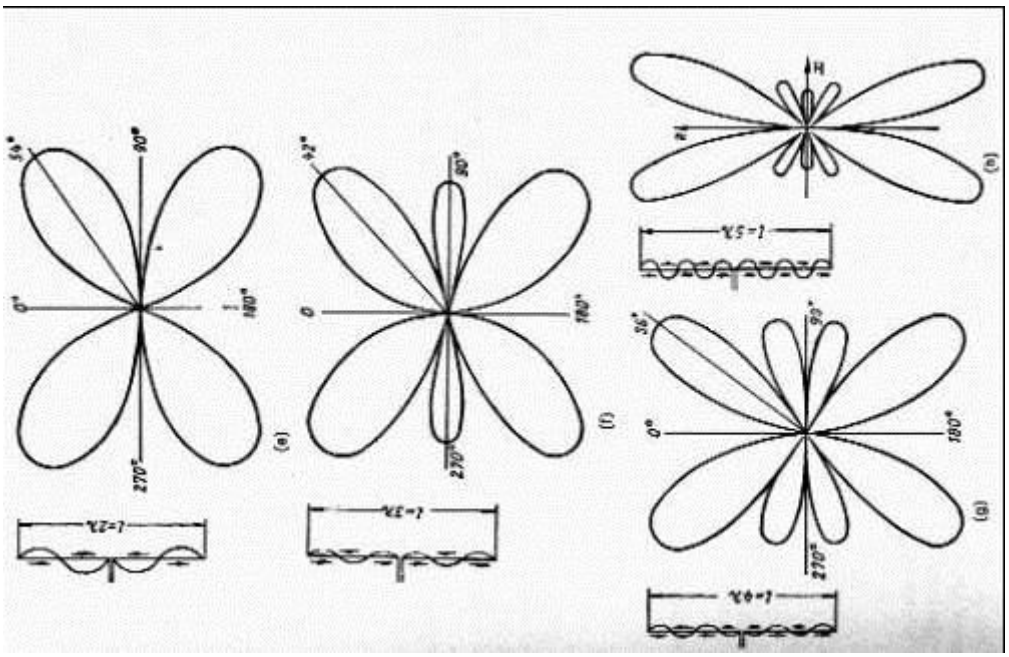


Figura 5.7 Diagramas de directividad, e) l = 2λ, f) l=3λ, g) l = 4 λ, h) l = 5 λ.

Si l = 2 λ (Fig.5.7e), las secciones con corriente en sentido opuesto son iguales en longitud, y por ello la radiación cesa completamente en el plano ecuatorial, al tiempo que a cada semiesfera corresponden dos lóbulos en el diagrama de directividad.
Si l = 3 λ (Fig.5.7f),la correspondencia entre las longitudes de las secciones del dipolo con diferentes sentidos de corrientes en el mismo caso que con l = 1.5 λ, y el diagrama de directividad tiene, en ambos casos, aproximadamente el mismo comportamiento.

Si l = 4 λ (Fig.5.7g), en cada semiesfera aparecen cuatro lóbulos del diagrama de directividad, no existiendo además radiación en el plano ecuatorial, pues las secciones del dipolo con corrientes en sentidos opuestos son iguales.

Si l = 5 λ (Fig.5.7h), no se observa esta igualdad, y uno de los lóbulos del diagrama de directividad se encuentra en el plano ecuatorial; además, a cada semiesfera corresponden otros cuatro lóbulos del diagrama.

Si se quiere que el dipolo simétrico no sea direccional en el plano ecuatorial, se obtiene su diagrama espacial haciendo girar el diagrama de directividad, en el plano meridional, con respecto al eje del dipolo

Conclusiones.

1. Un dipolo simétrico de cualquier longitud no irradia a largo de su eje, pues sus secciones elementales –equivalentes a dipolos elementales- no crean radiación en esa dirección.
2. Un aumento de la longitud de un dipolo por encima de λ va acompañado de una disminución de la radiación en el plano ecuatorial debido a que aparecen secciones del dipolo con sentidos opuestos de corriente.

3. Si l = 2λ, 4λ, 6λ,...., desaparece por completo la radiación en el plano ecuatorial debido a la igualdad de las longitudes de las secciones cuya corriente va en sentidos opuestos.
4. Si l = si l = λ, 2λ, 3λ, 4λ, 5λ...., el numero de los lóbulos del diagrama de directividad que corresponden a cada semiesfera es igual al numero de longitudes de onda que se disponen a lo largo del dipolo.
5. A medida que aumenta la longitud de un dipolo, la dirección del máximo del lóbulo principal va alejándose cada vez más del plano ecuatorial y acercándose al eje del dipolo.

5.1.8 Dipolo con interferencia de un diedro de 90˚

Se colocara el dipolo simétrico lo pondremos a una distancia a = λ/2, de un diedro de dos planos conductores perfectos, que forman un ángulo de 90˚. El dipolo es paralelo al eje z, y los planos están situados en Φ = π/4, Φ = -π/4.

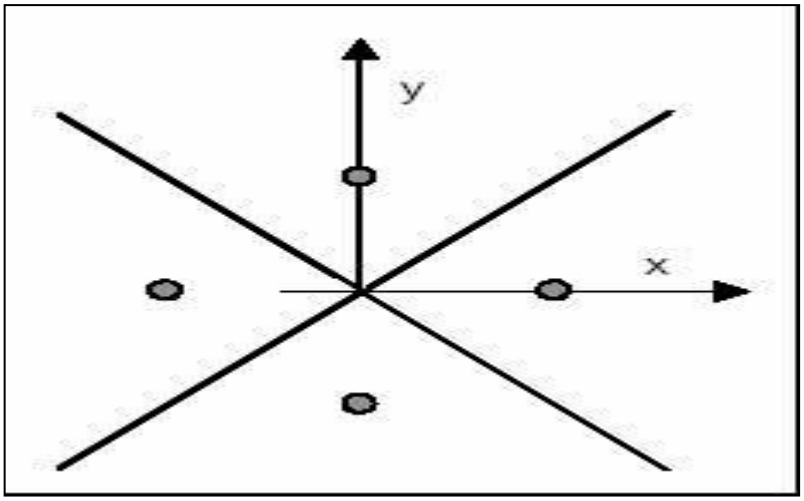


Figura 5.8 Representación del dipolo con un diedro de 90˚.

5.1.8.1 Vector de radiación

El vector de radiación de un dipolo alineado según el eje z es:

$$N_0 = \vec{Z}\,\beta I\left(\frac{\cos(\beta_Z H)-\cos\beta H}{\beta^2-\beta_Z^2}\right) \qquad \text{(Ec. 5.1.12)}$$

Los planos de masa se pueden analizar teniendo en cuenta las tres imágenes que aparecen

$$\vec{N} = \vec{N}_0\left(e^{j\beta_x a}-e^{j\beta_y a}+e^{-j\beta_x a}-e^{-j\beta_y a}\right) \qquad \text{(Ec. 5.1.13)}$$

$$\vec{N} = \vec{N}_0\,2\left(\cos(\beta_x a)-\cos(\beta_y a)\right) \qquad \text{(Ec. 5.1.14)}$$

En la dirección del eje x

$$\beta_x = \beta sen\theta \cos\phi = \beta \quad \text{(Ec. 5.1.15)}$$

$$\beta_y = \beta sen\theta \cos\phi = 0 \quad \text{(Ec. 5.1.16)}$$

$$\vec{N} = \vec{N}_0\, 2\left(\cos(\beta a) - 1\right) = \vec{N}_0\, 2\left(\cos\left(\frac{2\pi}{\lambda}\frac{\lambda}{2}\right) - 1\right) = -\vec{4}N_0 \quad \text{(Ec.5.1.17)}$$

5.1.8.2 Ecuación del dipolo con interferencia (Diedro).

Los campos radiados se pueden calcular a partir del potencial vector

$$A_z = \frac{\mu I_m e^{-j\beta r}}{2\pi\beta r}\left[\frac{\cos(\beta H\cos\theta) - \cos\beta H}{sen^2\theta}\right] 2\left(\cos(\beta_x a) - \cos(\beta_y a)\right) \quad \text{(Ec. 5.1.18)}$$

En coordenadas esféricas es[7]

$$A_r = A_Z\cos\theta \qquad A_\theta = -A_z sen\theta \qquad A_\phi = 0$$

$$E_\theta = \frac{j60 I_m e^{-j\beta r}}{r}\left[\frac{\cos\left(\frac{2\pi H}{\lambda}\cos\theta\right) - \cos\frac{2\pi H}{\lambda}}{sen\theta}\right] 2\left(\cos(\beta_x a) - \cos(\beta_y a)\right) \quad \text{(Ec.5.1.19)}$$

$$\vec{H}_\phi = \frac{E_\theta}{\eta} \quad \text{(Ec. 5.1.20)}$$

5.1.8.3 Diagrama de radiación del dipolo con interferencia.

El Plano E es el definido por la dirección de máxima radiación (eje x) y el campo eléctrico en dicha dirección (polarización vertical). Por lo tanto el plano E es el XZ. El plano H es el XY. En la Figura 5.9 se observan los diagramas correspondientes al plano E y H del dipolo, de la interferencia debida a las cuatro antenas y el producto de ambos. También se puede ver el diagrama tridimensional del dipolo, el factor de interferencia y el total.

Hay que notar que en realidad el campo sólo es diferente de cero en el interior del diedro. Por lo tanto los resultados para el campo total deben limitarse al sector correspondiente.

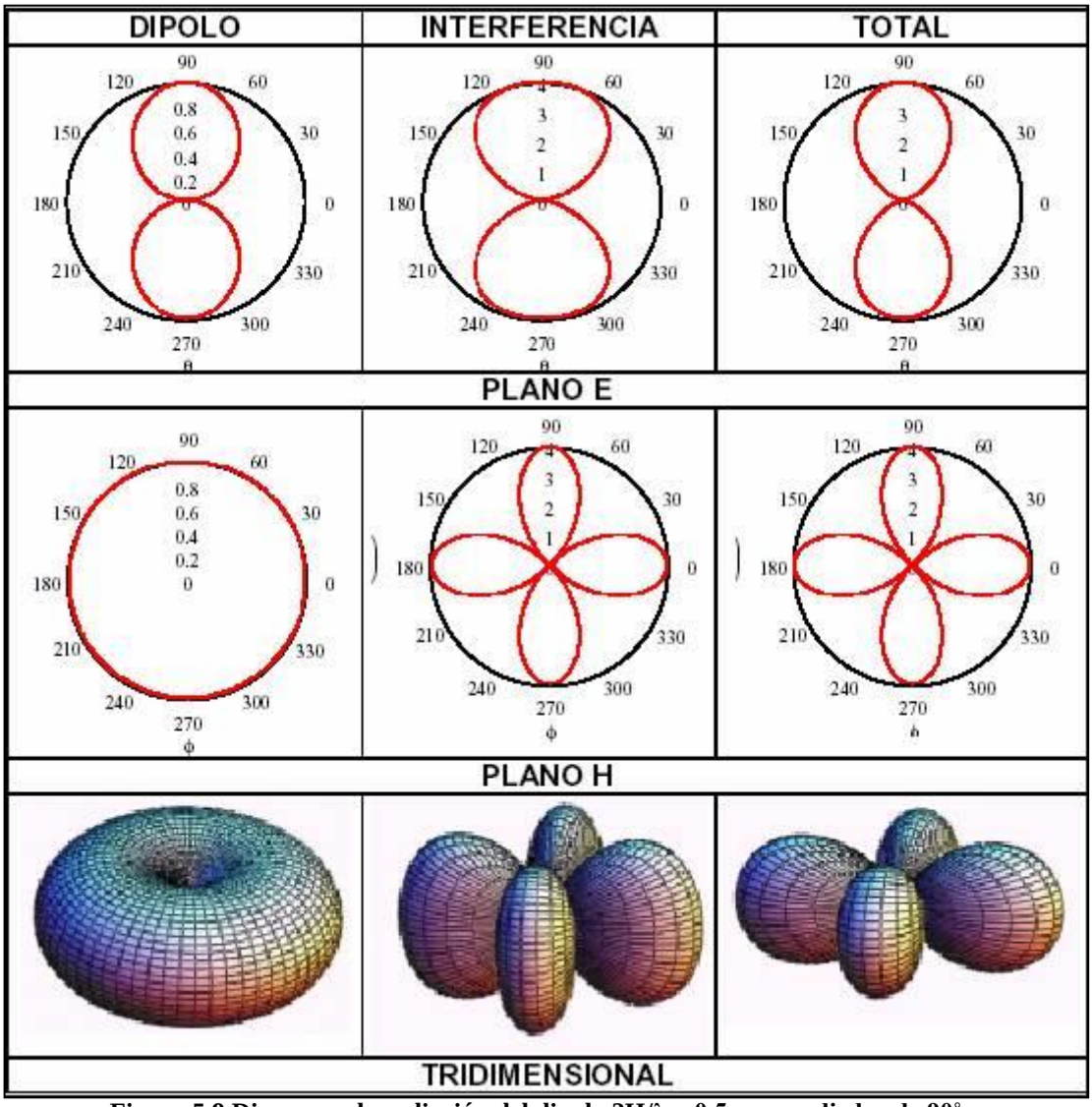


Figura 5.9 Diagrama de radiación del dipolo 2H/λ = 0.5 con un diedro de 90˚.

5.2 AGRUPACIONES DE ANTENAS.

5.2.1. Introducción.

Un arreglo es una antena compuesta por un número de radiadores idénticos ordenados regularmente y alimentados para obtener un diagrama de radiación predefinido.

Hay diferentes tipos de arreglos, los arreglos lineales tienen los elementos dispuestos sobre una línea. Los arreglos planos son agrupaciones bidimensionales cuyos elementos están sobre un plano. Los arreglos conformados tienen las antenas sobre una superficie curva, como por ejemplo el fuselaje de un avión.

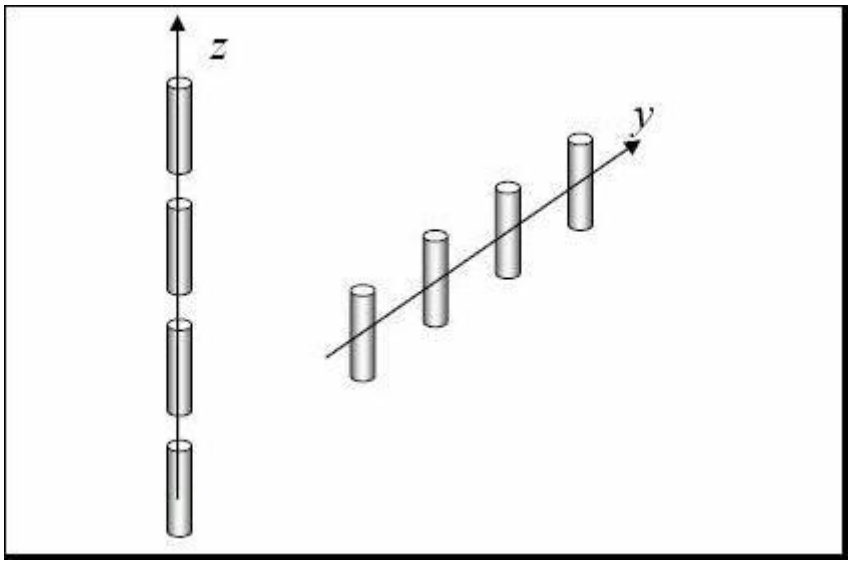


Figura 5.10 Agrupaciones de antenas.

Los arreglos tienen la ventaja de que se puede controlar la amplitud de las corrientes y la fase de cada elemento, modificando la forma del diagrama de radiación. Además se puede conseguir que los parámetros de antena dependan de la señal recibida a través de circuitos asociados a los elementos radiantes, como en el caso de los arreglos adaptativos.

5.2.2. Arreglo Lineal.

Los arreglos lineales de antenas son aquellos en los que los elementos del arreglo se encuentran ubicados sobre una línea recta, con separación uniforme o no uniforme. En la figura 5.11 se ilustra las variables dimensionales de un arreglo lineal genérico, en el cual la intensidad de campo eléctrico en la zona lejana producido por el arreglo se puede escribir como:

$$E_\theta \approx E_{\theta 1} \cdot AF \qquad \text{(Ec. 5.1.21)}$$

Donde $E_{\theta 1}$ es el campo eléctrico producido por la antena 1. AF se conoce como factor de arreglo y esta dado por[9]

$$AF = \sum_{k=1}^{N} a_k e^{j(kd_k \cos\theta + \beta_k)} \qquad \text{(Ec. 5.1.22)}$$

Dentro de los arreglos lineales de separación uniforme, existen desarrollos clásicos donde se maneja la amplitud uniforme y fase progresiva.

El factor de arreglos de arreglos con amplitud uniforme y fase progresiva se puede obtener a partir de la Ec (5.1.22) haciendo

$$d_1 = 0, \quad d_2 = d, \quad d_3 = 2d, \quad \cdots, d_N = (N-1)d$$

$$a_1 = a_2 = \cdots = a_N = 1$$

$$\beta_1 = 0, \quad \beta_2 = \beta, \quad \beta_3 = 2\beta, \quad \cdots, \quad \beta_N = (N-1)\beta$$

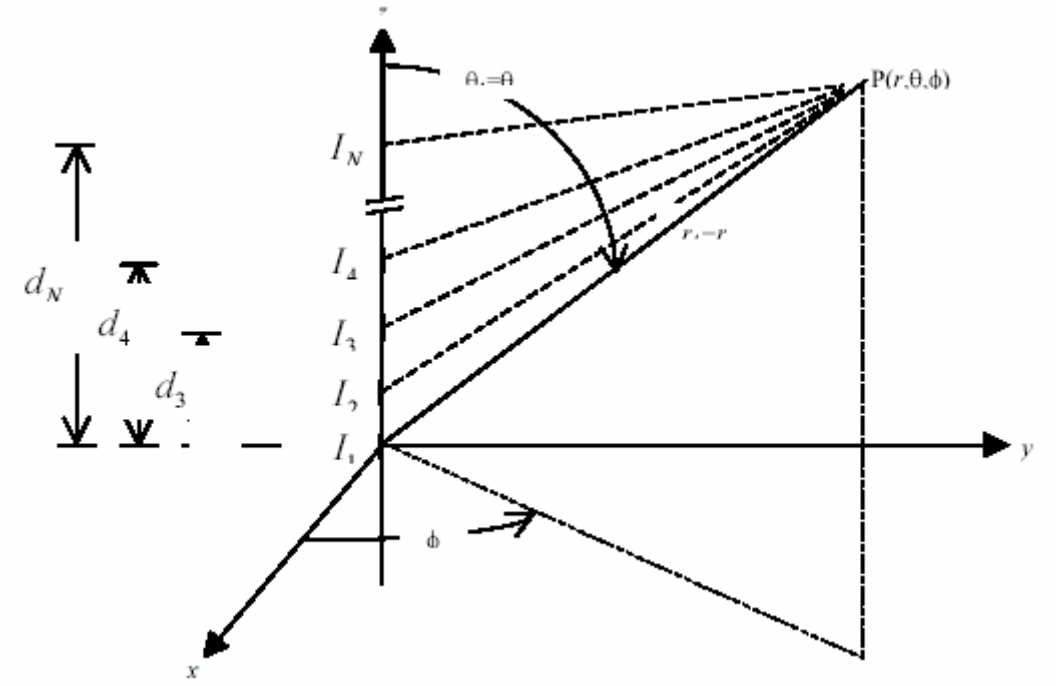


Figura 5.11 Disposición física de un arreglo lineal de antenas.

y dividiendo entre el número de elementos, se obtiene el factor de arreglo normalizado

$$AF_n = \frac{1}{N}\sum_{n=1}^{N} e^{j(n-1)\psi} \qquad \text{(Ec. 5.1.23)}$$

donde

$$\psi = kd\cos\theta + \beta \qquad \text{(Ec. 5.1.24)}$$

d es la distancia entre los elementos β es la fase progresiva. Es este caso, el numero de variables de diseño se reduce a 3 (numero de elementos, su separación, y su fase progresiva). Es posible demostrar que la Ec (5.1.23) se puede escribir como [9]

$$AF_N = \frac{e^{j(N-1)\psi/2}}{N}\frac{sen\,N\psi/2}{sen\,\psi/2} \neq e^{j(N-1)\psi/2}\frac{sen\,N\psi/2}{N\psi/2} \qquad \text{(Ec. 5.1.25)}$$

donde la aproximación es aplicable cuando ψ/2 <<1.

La dirección del lóbulo mayor se da en

$$\theta_{1,2} = \cos^{-1}\left(\frac{-\beta \pm \frac{2.782}{N}}{kd}\right) \qquad \text{(Ec. 5.1.26)}$$

y el lóbulo latera se presenta en

$$\theta_3 \approx \cos^{-1}\left(\frac{\frac{3\pi}{N} - \beta}{kd}\right) \qquad \text{(Ec. 5.1.27)}$$

cuya amplitud corresponde a 13.6 dB por debajo del lóbulo mayor.

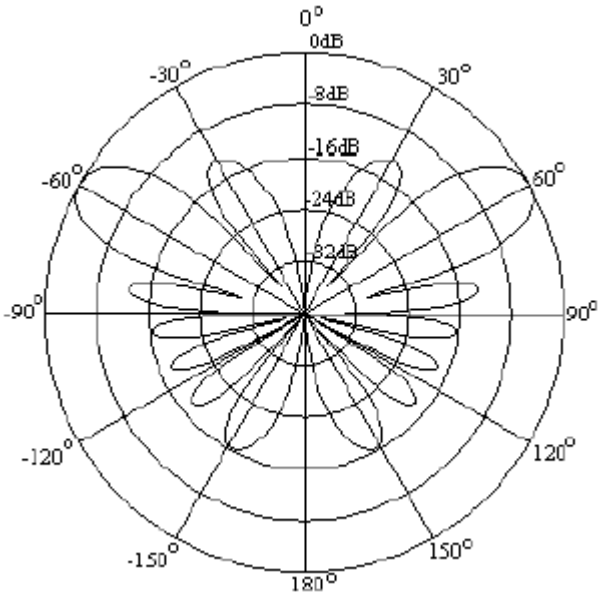


Figura 5.12 Factor de arreglo de un arreglo de amplitud uniforme y fase progresiva.

5.2.2.1 Antena Yagi-Uda

La Yagi-Uda, inventada en Japón en 1926 por S. Uda y dada a conocer internacionalmente poco después por H. Yagi. La característica más importante de esta antena es su simplicidad: consiste en un dipolo resonante de media onda y varios dipolos cortocircuitados: el primero, conocido como elemento activo, es conectado por

su centro directamente a una línea de transmisión, mientras que los otros (elementos parásitos), actúan como radiadores cuyas corrientes son inducidas por acoplamiento mutuo y están dispuestos todos en un mismo plano, sobre un soporte común (boom) y paralelos entre sí [8].

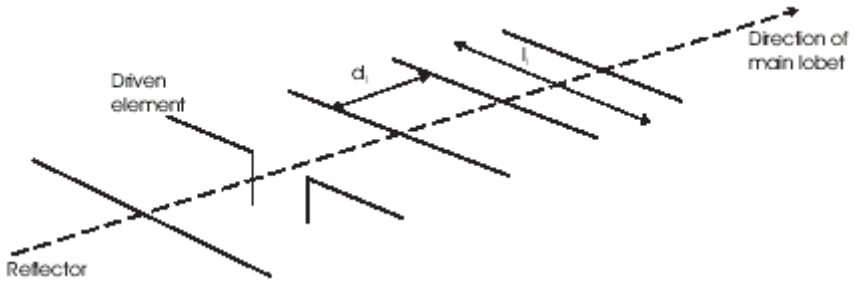


Figura 5.13 Antena Yagi-Uda

5.2.2.2. Ecuaciones Básicas

Para pequeños diámetros (con respecto a la distancia a la fuente **z'**) de los elementos que conforman la antena Yagi-Uda, la corriente (I_n) sobre cada uno de ellos se puede aproximar por la siguiente serie finita de Fourier:

$$I_n(z') = \sum_{m=1}^{M} I_{mn} \cos\left[(2m-1)\frac{\pi z'}{l_n}\right],\ m = 1,2,3,4,\ldots$$

(Ec. 5.1.28)

donde I_{mn} representa el coeficiente de corriente compleja de modo m (componente armónica de la corriente para cada elemento) sobre el elemento n y l_n la correspondiente longitud del elemento.

Usando la formulación de los potenciales retardados [8], se puede demostrar que el campo eléctrico radiado por todos los elementos de la antena Yagi-Uda, cada uno con una corriente dada por (1), es:

$$\vec{E}_\theta = \sum_{n=1}^{N} \vec{E}_{\theta_n} = -jn\frac{e^{-jkr}}{4r}\vec{F}_T(\theta,\phi),$$

(Ec. 5.1.29)

donde,

$$\left|\vec{F}_T(\theta,\phi)\right| = \operatorname{sen}\theta \sum_{n=1}^{N}\left\{ I_n e^{j\Psi_n}\left[\sum_{m=1}^{M}(-1)^m \frac{(2m-1)I_{mn}\cos\left(\frac{\pi l_n}{\lambda}\cos\theta\right)}{(2m-1)^2 - \left(\frac{2l_n}{\lambda}\cos\theta\right)^2}\right]\right\}$$

(Ec. 5.1.30)

proporciona el diagrama de radiación y

$$\bar{\Psi}_n \equiv k\left(\bar{x}_n \operatorname{sen}\theta\cos\phi + \bar{y}_n \operatorname{sen}\theta \operatorname{sen}\phi + \bar{z}_n\cos\theta\right)$$

(Ec. 5.1.31)

siendo λ la longitud de onda, k el número de onda, θ y ø las coordenadas altazimutales y $j \equiv \sqrt{-1}$.

5.2.3. Arreglos Planos.

Los arreglos planos son aquellos en los que los elementos del arreglo se encuentran ubicados en un plano, existen varias configuraciones que pueden ser obtenidas; por ejemplo, arreglos rectangulares, circulares, o en cruz.

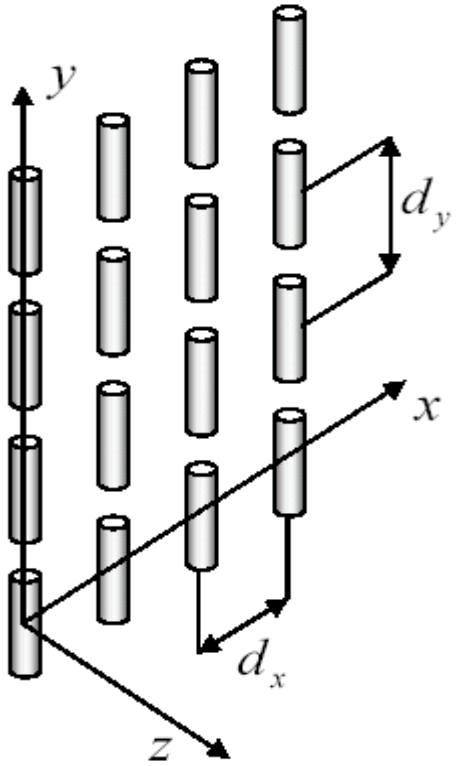


Figura 5.14 Arreglos planos de antenas.

Los arreglos planos son mucho mas versátiles que los arreglos lineales [10] ya que posee mayor numero de parámetros de control, permitiendo la obtención de patrones mas simétricos con lóbulos laterales mas pequeños y facilitando el desplazamiento del haz hacia cualquier punto del espacio.

Estas características convierten a los arreglos planos en antenas ideales para aplicaciones tales como radar, antenas inteligentes aplicadas a sistemas modernos de comunicaciones radioastronomía, telemetría, etc.

Un arreglo plano puede estructurarse considerando un arreglo lineal de M elementos colocados, por ejemplo, a lo largo del eje x, y posteriormente repetir N de tales arreglos

a lo largo de la dirección y como como se muestra en la figura 5.14. de esta forma cada elemento en el arreglo original en la dirección x estará espaciado por la distancia d_x y una ase progresiva β_x mientras en la dirección y cada arreglo o elemento tendrá una separación d_y y una fase progresiva β_y. En esta forma quedaría un arreglo de tipo rectangular. El factor de arreglo para este arreglo plano puede expresarse como

$$AF = \sum_{n=1}^{N} I_{1n} \left[\sum_{m=1}^{M} I_{m1} e^{j(m-1)(kd_x \operatorname{sen}\theta \cos\varphi + \beta_x)} \right] \cdot e^{j(n-1)(kd_y \operatorname{sen}\theta \cos\varphi + \beta_y)} \qquad \text{(Ec. 5.1.32)}$$

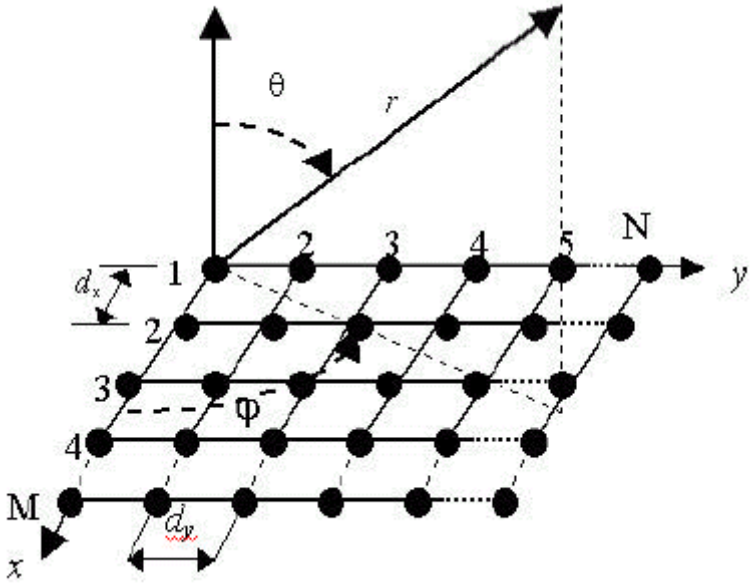


Figura 5.15 Disposición física de un arreglo plano rectangular de antenas.

De la Ec. (5.1.32) se observa que el factor de arreglo rectangular es el producto de los factores de arreglo de los arreglos en las direcciones *x* y *y*.

Tomando los coeficientes de excitación en amplitud de los elementos del arreglo en la dirección *y* proporcionales a la amplitud de los elementos en *x*, la amplitud del elemento (m,n)-esimo puede escribirse como

$$I_{mn} = I_{m1} I_{1n}$$

para el caso de un arreglo uniforme ($I_{mn} = I_0$) el factor de arreglo normalizado puede expresarse como

$$AF_n(\theta,\varphi)=\left\{\frac{1}{M}\frac{\operatorname{sen}\left(\frac{M}{2}\psi_x\right)}{\operatorname{sen}\left(\frac{\psi_x}{2}\right)}\right\}\left\{\frac{1}{N}\frac{\operatorname{sen}\left(\frac{M}{2}\psi_y\right)}{\operatorname{sen}\left(\frac{\psi_y}{2}\right)}\right\} \qquad \text{(Ec. 5.1.33)}$$

donde

$$\psi_x = kd_x \operatorname{sen}\theta\cos\varphi + \beta_x$$

$$\psi_y = kd_y \operatorname{sen}\theta\operatorname{sen}\varphi + \beta_y$$

5.2.3.1 Cruz de Mills

La cruz de Mills consiste de dos arreglos lineales horizontales, colocados perpendicularmente y ubicados simétricamente respecto del punto central. Cuando los elementos elegidos son dipolos orientados este-oeste, se obtiene el arreglo presentado en la figura 5.16.

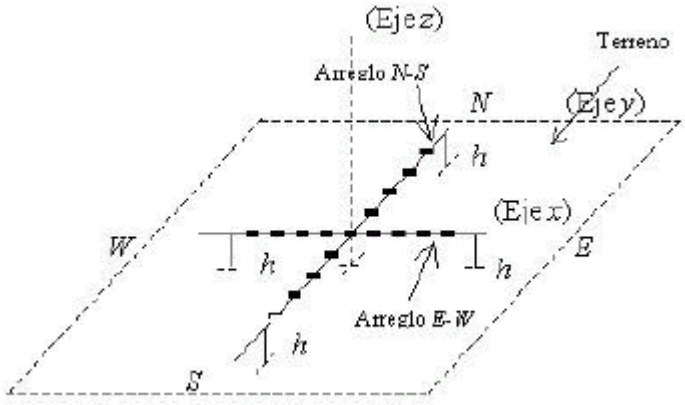


Figura 5.16 Cruz de Mills con dipolos en la dirección este-oeste sobre un terreno conductor.

El campo total producido por los arreglos presentados en la figura 5.15 es igual a la suma de los campos individuales de cada arreglo, es decir

$$E_T = E_{NS} + E_{EW} \qquad \text{(Ec. 5.1.34)}$$

donde

$$E_{NS} = E_{e1} AF_{NS} \qquad \text{(Ec. 5.1.35)}$$

$$E_{EW} = E_{e2} AF_{EW} \qquad \text{(Ec. 5.1.36)}$$

donde E_{e1} y E_{e2} son los campos producidos por los elementos utilizados en cada arreglo. En el caso de utilizar un mismo tipo de elementos $E_{e1} = E_{e2} = E_e$, entonces el campo total será

$$E_T = E_e\left(AF_{NS} + AF_{EW}\right)$$

La configuración global involucra la proximidad del terreno por lo que su efecto se incorpora considerando cada elemento como un arreglo de dos elementos (fuente virtual), así el campo total será:

$$E_T = E_d\, AF_t\left(AF_{NS} + AF_{EW}\right) \qquad \text{(Ec. 5.1.37)}$$

donde E_d es el campo producido por cada dipolo horizontal y AF_t es el factor de arreglo debido al efecto del terreno, este se puede calcular como [10]:

$$AF_t = ja_1 sen\left(kh\cos\theta + \beta_1\right) \qquad \text{(Ec. 5.1.38)}$$

Los parámetros de diseño en el arreglo de la figura 5.15 son: la longitud del dipolo, la altura del arreglo respecto al terreno, las distancias entre los elementos y la excitación de los mismos.

5.3 Antena Parabólica.

5.3.1. Definición de Antena Parabólica.

Una antena parabólica consiste en un alimentador, generalmente una Antena de bocina que ilumina una superficie reflectora con una onda radio. El objetivo es radiar una onda plana en una determinada dirección en la que los desfases entre puntos del plano sean nulos. Por ello se utiliza un reflector de tipo parabólico donde el alimentador se sitúa en su foco.
De esta forma la distancia recorrida por cualquier rayo que sale del foco tras reflejarse es la misma y el desfase respecto a otro rayo nulo. El esquema se puede observar en la Figura 5.3.1.

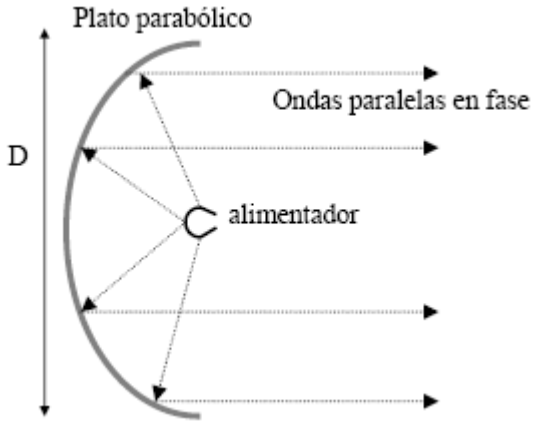


Figura 5.3.1 Esquema de una antena Parabólica.

Las antenas parabólicas son habitualmente analizadas empleando las técnicas siguientes:

- Óptica geométrica.
- Óptica Física
- GTD.

5.3.2 Óptica Geométrica.

La óptica geométrica permite calcular los campos en la apertura, los campos lejanos se obtienen usando los principios de equivalencia. Constituye una buena y sencilla aproximación para calcular el lóbulo principal y primeros lóbulos secundarios.
Los rayos de óptica geométrica, son de familia de curvas que no son normales a las superficies equifasicas de los frentes de onda. Estos rayos permiten definir tubos de propagación que cumplen la ley de conservación de energía si el medio no tiene perdidas. Las condiciones de reflexión se cumplen estrictamente en el punto de reflexión, pero la onda incidente y la onda reflejada dependen del alimentador utilizado y de la forma de la superficie reflejada.

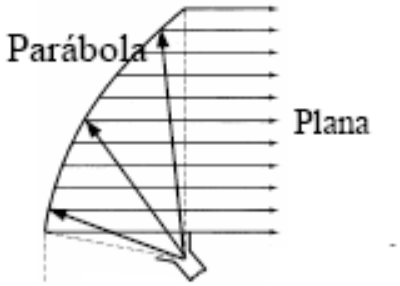


Figura 5.3.2 Ejemplo de la transformación de ondas con superficies cuadricas.

La óptica geométrica estudia la propagación de las ondas electromagnéticas apoyándose en las leyes puramente geométricas.
En régimen permanente sinusoidal, con campos de la forma:

$$\bar{E}(x,y,z,t) = \bar{E}_o(x,y,z)\, e^{j(\omega t - k\psi(x,y,z))} \qquad \text{(Ec. 5.3.1)}$$

Haciendo en las ecuaciones de Maxwell $\lambda \to 0$ se obtiene

$$|\nabla\psi(x,y,z)| = n(x,y,z) \qquad \text{(Ec. 5.3.2)}$$

Donde n es el índice de refracción

$$n = \sqrt{\mu_r \varepsilon_r} \qquad \text{(Ec. 5.3.3)}$$

Siendo Ψ =constante para superficies equifasicas

$$\vec{E}_0 \perp \nabla\psi \quad \vec{H}_0 \perp \nabla\psi \quad \langle\vec{S}\rangle // \nabla\psi \qquad \text{(Ec. 5.3.4)}$$

En medios homogéneos con n= cte, como ocurre en el estudio de reflectores, los rayos son rectilíneos y los campos cumplen localmente las mismas propiedades de las ondas planas. Los resultados obtenidos serán tanto más precisos cuando mayores sean los tamaños eléctricos y los radios de curvatura de los reflectores.

5.3.2.1 Principio de Fermat.

La longitud del camino óptico recorrido por un rayo estacionario, esto es, su derivada primera es nula.
La longitud del camino óptico se define en función del índice de refracción del medio n y del camino físico recorrido L como:

$$L_{optica} = \int_L n\, dl = \int_L \sqrt{\varepsilon_r \mu_r}\, dl \qquad \text{(Ec. 5.3.5)}$$

Habitualmente la estacionareidad coincide con un mínimo local de la longitud óptica. El Principio de Fermat permite diseñar lentes de índice de refracción variable, como la de Lunemberg y obtener los puntos de reflexión sobre superficies reflectoras.

5.3.2.2 Leyes de Snell.

Son derivadas del principio de Fermat sobre los rayos que inciden en el plano. En el caso del rayo incidente reflejado y la normal a la superficie en el punto de reflexión**.** Como se observa en la figura 5.3.3 el ángulo de incidencia y el de reflexión (medidos respecto ala normal) son iguales:

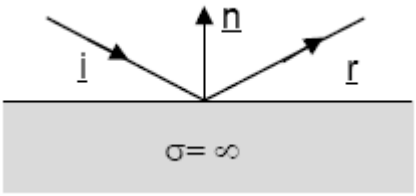


Figura 5.3.3 Reflexión de Ondas.

De donde obtienen las ecuaciones siguientes:

$$\hat{n} \times \left(\vec{r} - \vec{i} \right) = 0 \qquad \text{(Ec. 5.3.6)}$$

$$\hat{n} \cdot \left(\vec{r} + \vec{i} \right) = 0 \qquad \text{(Ec. 5.3.7)}$$

$$\vec{r} = \vec{i} - 2\left(\vec{i} \cdot \hat{n} \right)\hat{n} \qquad \text{(Ec. 5.3.8)}$$

En el caso de las ondas refractadas (figura 5.3.4), la relación entre los senos de los ángulos de incidencia y refracción es proporcional a los índices de refracción.

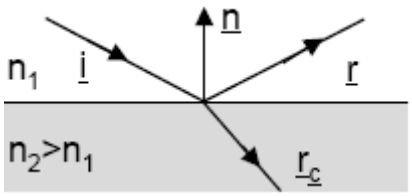


Figura 5.3.4 Refracción de Ondas.

De donde se obtiene la ecuación.

$$\hat{n} \times \left(n_2 \hat{r}_c - n_1 \hat{i} \right) = 0 \qquad \text{(Ec. 5.3.9)}$$

Para un reflector parabólico, cada punto de incidencia se aproxima a la superficie reflectora por un plano tangente conductor perfecto, de modo que se cumplen las leyes de Snell y la condición de contorno $E_{Total}|_{Tangente} = 0$.

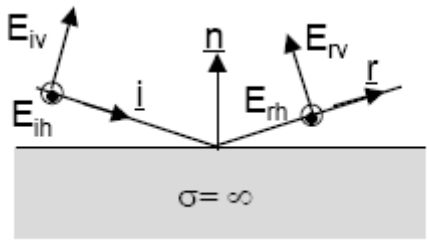


Figura 5.3.5 Ondas en el Reflector Parabólico.

$$\begin{bmatrix} E_{rv} \\ E_{rh} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} E_{iv} \\ E_{ih} \end{bmatrix}$$

Resolviendo la matriz se obtiene

$$\vec{E}_r = 2\left(\hat{n} \cdot \vec{E}_i\right)\hat{n} - \vec{E}_i \qquad \left(\vec{E}_i \perp \hat{i}\right)$$

$$\left|\vec{E}_r\right| = \left|\vec{E}_i\right|$$

(Ec. 5.3.10)

Las Condiciones anteriores se cumplen estrictamente en el punto de reflexion, pero la onda incidente y la onda reflejada dependen del alimentador utilizado y de la forma de la superficie reflectora. Los reflectores enfocados, basados en cuadricas las ondas que se manejan son:

- Ondas Esféricas. (pequeño alimentador = Fuente puntual)

$$E = E_0 \frac{e^{-jkr}}{r}$$

(Ec. 5.3.11)

- Ondas Cilíndricas(Campo Próximo de una fuente lineal)

$$E = E_0 \sqrt{\frac{1}{\rho}}\, e^{-jk\rho}$$

(Ec. 5.3.12)

- Ondas localmente planas(Onda colimada por reflector parabolico)

$$E = E_0\, e^{-jkz}$$

(Ec. 5.3.13)

5.3.3 Óptica Física.

Esta calcula las corrientes inducidas sobre la superficie metálica iluminada por el campo incidente.

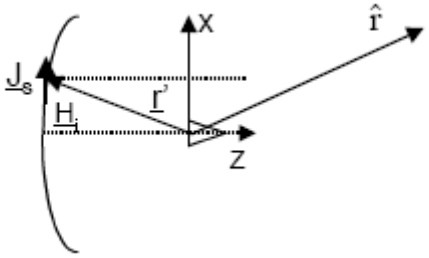


Figura 5.3.6 Óptica física.

Se define la densidad de corriente, bajo las condiciones que se posee un conductor perfecto y un radio de curvatura infinito.

$$\vec{J}_s = 2\hat{n} \times \vec{H}_i = \frac{2}{\eta}\left(\hat{n} \times \left(\hat{i} \times \vec{E}_i\right)\right) \qquad \text{(Ec. 5.3.14)}$$

El campo radiado por esas corrientes vale.

$$\vec{E}(\vec{r}) = \frac{-j\omega\mu}{4\pi}\frac{e^{-jkr}}{r}\iint_S\left[\vec{J}_s(\vec{r}') - \left(\vec{J}_s(\vec{r}')\cdot\hat{r}\right)\hat{r}\right]e^{jkr\cdot r'}dS' \qquad \text{(Ec. 5.3.15)}$$

Donde S es la superficie iluminada por el reflector.

5.3.3.1 Integración sobre la Apertura.

Conocidas las corrientes J_S en la superficie del reflector una función equivalente en la apertura $f_r(r, \varphi)$ se puede obtener utilizando el jacobiano como se observa en la figura 5.3.7:

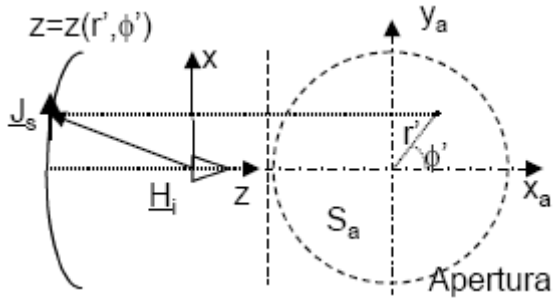


Figura 5.3.7 Corrientes incidentes en el Reflector.

De donde se obtiene la función $f_r(r, \varphi)$.

$$\vec{f}(r',\phi') = \vec{J}_s(r',\phi',z)\sqrt{1+\left(\frac{\partial z}{\partial r'}\right)^2+\left(\frac{\partial z}{\partial \phi'}\right)^2} = \vec{J}_s(r',\phi',z)\sqrt{1+\left(\frac{\partial z}{\partial r'}\right)^2} \qquad \text{(Ec. 5.3.16)}$$

Para el caso de simetría de revolución. El campo radiado se calcula en función de $f_r(r', \varphi')$ como:

$$\vec{E}(\vec{r}) = \frac{-j\omega\mu}{4\pi}\frac{e^{-jkr}}{r}\iint_{S_a}\left[\vec{f} - (\vec{f}\cdot\hat{r})\hat{r}\right]e^{jk\hat{r}\cdot\vec{r}'}dS'_a \qquad \text{(Ec. 5.3.17)}$$

$f_r(r', \varphi')$ coincide con los campos de apertura calculados usando óptica geométrica.
Los campos radiados por una antena que incluya un reflector dependerán por una parte de las características de dicha antena, (Ancho de haz, nivel de lóbulo principal a secundario, Polarizacion, etc.), y por otra parte de las características geométricas del reflector (distancia focal, diámetro).
La densidad de potencia incidente en el reflector se puede calcular utilizando la ecuación de transmisión.

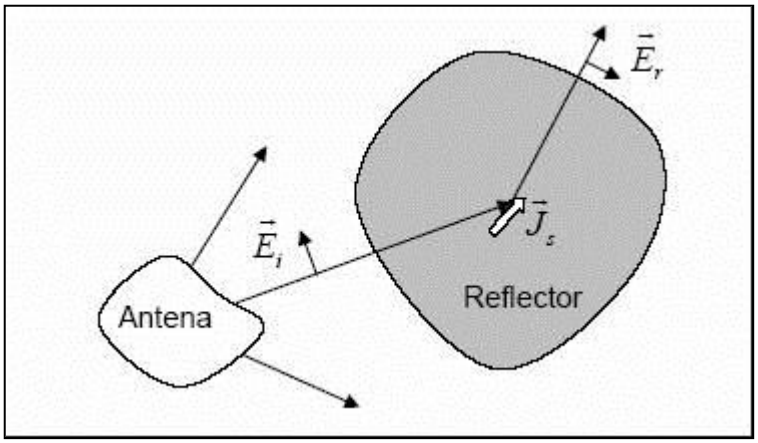


Figura 5.3.8 Densidad de potencia en el reflector.

$$P(\theta',\phi') = \frac{|E_i|^2}{\eta} = \frac{W_t}{4\pi r^2}D(\theta',\phi') \qquad \text{(Ec. 5.3.18)}$$

El Campo incidente en el reflector tiene una orientación vertical, dependiendo de la Polarizacion, y una fase que incluye el camino recorrido por la onda esférica.

$$\vec{E}_i = \frac{e^{-jkr}}{r}\left(\frac{\eta W_t D(\theta',\phi')}{4\pi}\right)^{\frac{1}{2}}\hat{e}_i \qquad \text{(Ec.5.3.19)}$$

Para calcular el campo eléctrico reflejado, se debe cumplir la condición de campo eléctrico tangencial, suma del incidente y reflejado deben ser cero.

$$\hat{n}\times\left(\vec{E}_i + \vec{E}_r\right) = 0 \qquad \text{(Ec.5.3.20)}$$

Los campos magnéticos están relacionados con el eléctrico a través de la impedancia de onda.

$$\vec{H}_i = \frac{\left(\hat{n}_i \times \vec{E}_i\right)}{\eta} \qquad \text{(Ec. 5.3.21)}$$

$$\vec{H}_r = \frac{\left(\hat{n}_r \times \vec{E}_r\right)}{\eta} \qquad \text{(Ec. 5.3.22)}$$

Las corrientes inducidas por el reflector se pueden calcular como:

$$\vec{J}_s = \hat{n} \times \left(\vec{H}_i + \vec{H}_r\right) \qquad \text{(Ec. 5.3.24)}$$

Para el cálculo de campos radiados es necesario realizar el calculo del vector de radiación de dichas corrientes.

$$\vec{N} = \int \vec{J}_s e^{j\vec{\beta}\cdot\vec{r}'} ds' \qquad \text{(Ec. 5.3.25)}$$

5.3.4 GTD.

Los anteriores métodos de análisis permiten simular con muy buena precisión el lóbulo principal y los lóbulos secundarios adyacentes. Para analizar la radiación lejana es necesario GTD.
La teoría geométrica de difracción (GTD) postula el campo dispersado como la suma de los:

- Campos de óptica Geométrica(Rayo directo y reflejado)
- Campos difractados por aristas y bordes.

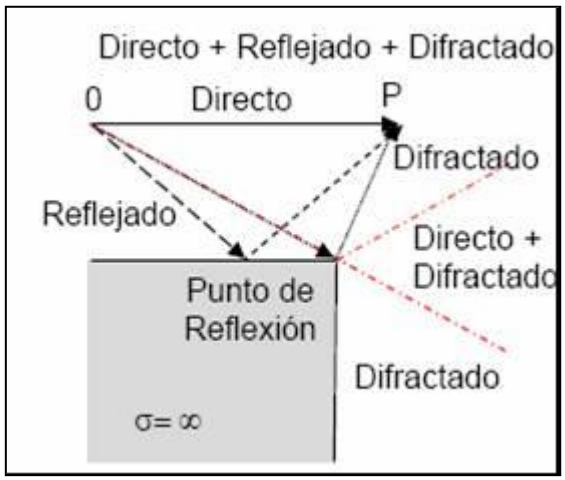


Figura 5.3.9 GTD.

5.3.4.1 Propiedades de los Campos Difractados.

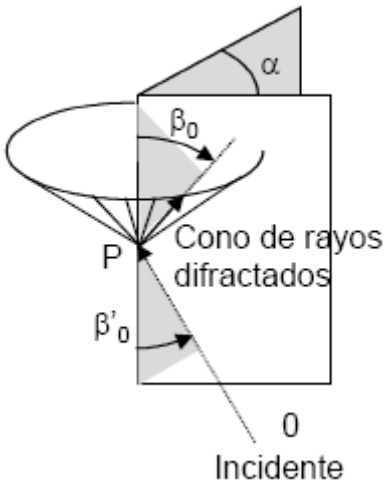


Figura 5.3.10 Rayos Difractados.

- Los Rayos difractados emergen radialmente de los bordes del reflector formando conos de rayos, centrados sobre las rectas tangentes a dicho borde, de acuerdo con la formulación de Keller, que establece que $\beta_{0=} \beta'_0$.

- Los puntos de difracción se calculan de acuerdo al principio de fermat.

- Los campos difractados se obtienen como producto de los campos incidentes por unos coeficientes de difracción, en función de β, α y de los ángulos que forman los rayos incidente y difractado en el plano de la arista.

5.3.5 Análisis de los Reflectores Parabólicos Como Aperturas.

Desde el foco hasta el reflector las ondas se propagan como ondas esféricas.

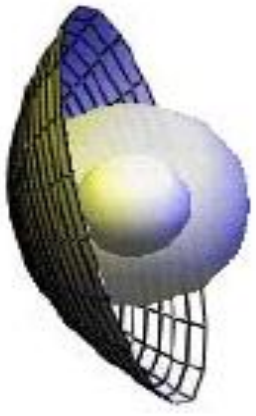


Figura 5.3.11 Propagación de la Onda Esférica en la Superficie Parabólica.

Una vez reflejadas en la superficie parabólica se propagan como ondas planas hasta la apertura (Ec. 5.3.13)

Figura 5.3.12 Propagación de Onda Plana en Superficie Parabólica.

La amplitud del campo en las ondas planas es constante, tan solo varia la fase. En ondas esféricas que se propagan entre el foco y el reflector, el modulo del campo será proporcional a la inversa de la distancia y al diagrama de campo de la antena situada en el foco.

$$|E_a| \propto \frac{\sqrt{D(\theta',\phi')}}{r} \qquad \text{(Ec. 5.3.25)}$$

La fase del campo es constante teniendo en cuenta las propiedades de la parábola.

$$fase(E_a) = -jk(R + z_a + z) = cte \qquad \text{(Ec. 5.3.26)}$$

La distribución de los campos en la apertura depende de los dos factores citados: diferencia de los caminos recorridos y atenuación debido al diagrama del alimentador. Las ondas que se propagan entre el foco y el reflector se comportan como ondas esféricas, y los campos varían en fase y en amplitud. Las ondas que se propagan desde el reflector hasta la apertura se comportan como ondas planas y tan solo se modifica la fase. Tomando como referencia el centro del reflector, la directividad en dicha dirección es Do y el camino recorrido es f. La atenuación total de los campos será:

$$\tau = 20\log\left(\frac{\sqrt{\frac{D(\theta',\phi')}{D_0}}}{\frac{f}{R}}\right) = 10\log\left(\frac{D(\theta',\phi')}{D_0}\right) + 20\log\left(\frac{R}{f}\right) \qquad \text{(Ec. 5.3.27)}$$

Teniendo en cuenta la ecuación parametrica de la parábola.

$$R = \frac{f}{\cos^2 \frac{\alpha}{2}}$$

(Ec. 5.3.28)

La atenuación de los campos en la apertura es debido a la diferencia de caminos y al diagrama.

$$\tau = \tau_1 + \tau_2 = 40 \log\left(\cos\frac{\alpha}{2}\right) + 10 \log\left(\frac{D(\theta', \phi')}{D_0}\right)$$

(Ec. 5.3.29)

Una vez conocida la distribución de campos en la apertura, es posible calcular los campos radiados, utilizando la teoría que ha sido desarrollada para aperturas circulares.
El valor de τ permite aproximar la distribución para una función uniforme mas una distribución tipo parabólica o parabólica al cuadrado.
Una apertura circular uniforme tiene un diagrama proporcional a la transformada de los campos en la apertura.

$$\iint_{s'} E_u(x', y') e^{j\vec{\beta}\cdot\vec{\rho}'} ds' = E_u\left(\pi a^2 2 \frac{J_1(ka \sin\theta)}{ka \sin\theta}\right)$$

(Ec. 5.3.30)

Si la apertura circular tiene simetría de revolución, el parámetro fundamental de análisis es la variación radial de los campos en la apertura.
Las propiedades del reflector parabólico dependen de los parámetros, distancia focal a diámetro de la apertura (f/2a). Un valor reducido equivale a un reflector con gran curvatura, mientras que valores superiores a uno suponen que el reflector esta mas cerca de un plano.

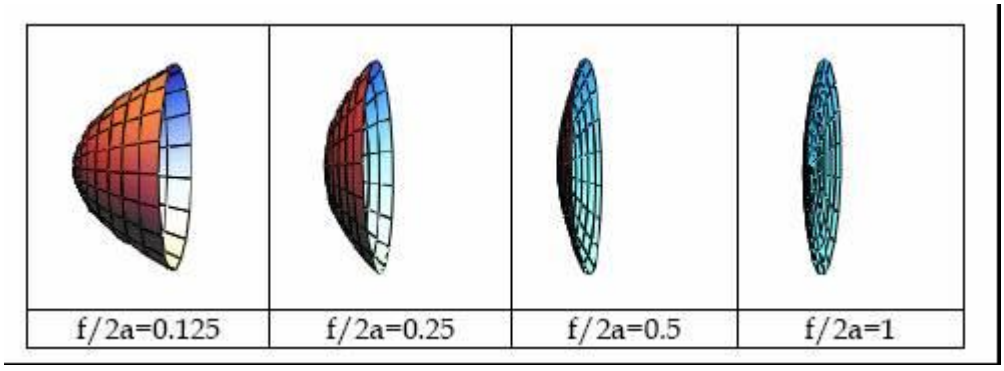


Figura 5.3.13 Curvaturas de Reflectores Parabólicos.

La relación que liga el ángulo máximo con el parámetro f / 2 a se pueden obtener a partir de las ecuaciones de la parábola.

$$\rho^2 = 4f(z + z_a)$$

(Ec. 5.3.31)

$$\rho = 2f \tan\frac{\alpha}{2}$$

(Ec.5.3.32)

Particularizando el extremo de la parábola resulta la relación entre los distintos parámetros de la curva.

$$\left(\frac{a}{2}\right)^2 = 4f\left(z + z_a\right) = 4f\,h \qquad \text{(Ec. 5.3.33)}$$

$$a = 2f\tan\frac{\beta}{2} \qquad \text{(Ec. 5.3.34)}$$

Las Relaciones que ligan las dimensiones geométricas con la relación f/D son.

$$h = \frac{1}{16\left(\frac{f}{2a}\right)^2} f \qquad \text{(Ec. 5.3.35)}$$

$$\tan\frac{\beta}{2} = \frac{1}{4\left(\frac{f}{2a}\right)} \qquad \text{(Ec. 5.3.36)}$$

5.3.6 Directividad de los reflectores Parabólicos.

Un reflector Parabólico se puede analizar como una apertura circular en la que conocemos el valor de la distribución de los campos. Los campos radiados si la polarizacion es circular serian.

$$E_\theta = j\frac{e^{-j\beta r}}{2\lambda r}\cos\phi\left(\frac{\eta}{Z_0}\cos\theta + 1\right)\iint_{s'} E\left(x', y'\right) e^{jk_x x'} e^{jk_y y'} dx'\,dy' \qquad \text{(Ec. 5.3.37)}$$

$$E_\phi = -j\frac{e^{-j\beta r}}{2\lambda r}\sin\phi\left(\frac{\eta}{Z_0} + \cos\theta\right)\iint_{s'} E\left(x', y'\right) e^{jk_x x'} e^{jk_y y'} dx'\,dy' \qquad \text{(Ec.5.3.38)}$$

La directividad se calcula como:

$$D_p = \frac{\frac{P_m}{W_t}}{\frac{}{4\pi r^2}} = \frac{4\pi r^2}{\eta}\frac{E_\theta^2 + E_\phi^2}{W_t} \qquad \text{(Ec. 5.3.39)}$$

$$D_p = \frac{4\pi}{\lambda^2}\frac{\left(\iint_{s'} E_a(x',y')ds'\right)^2}{\iint_{s'}\left|E_a(x',y')\right|^2 ds'} = \frac{4\pi}{\lambda^2}\frac{\left(\iint_{s'} E_a(x',y')\rho d\rho d\phi'\right)^2}{\iint_{s'}\left|E_a(x',y')\right|^2 \rho d\rho d\phi'}$$

$$D_p = \frac{4\pi}{\lambda^2}\pi a^2 \varepsilon_{il} \qquad \text{(Ec. 5.3.40)}$$

5.3.7 Eficiencia de Iluminación.

Los Campos de apertura son proporcionales al campo recorrido por las ondas y al diagrama de la antena situada en el foco. Si la antena tiene simetría de revolución.

$$\vec{E}_a = \frac{e^{-j\beta r}}{r}\left(\frac{\eta W_t D(\alpha)}{4\pi}\right)^{\frac{1}{2}} e^{-jk(x_a - x)}\hat{e}_r \qquad \text{(Ec. 5.3.41)}$$

$$E_a = E_0\frac{\sqrt{D(\alpha)}}{R} \qquad \text{(Ec. 5.3.42)}$$

Sustituyendo los valores de las ecuaciones parametricas de la parábola (Ec. 5.3.32) y (Ec. 5.3.28)

$$d\rho = \frac{f}{\cos^2\left(\frac{\alpha}{2}\right)}d\alpha \qquad \text{(Ec. 5.3.43)}$$

$$\eta_{il} = \frac{1}{\pi a^2}\frac{\left(\iint_{s'} E_a(x',y')\rho d\rho d\phi'\right)^2}{\iint_{s'}\left|E_a(x',y')\right|^2 \rho d\rho d\phi'} \qquad \text{(Ec. 5.3.44)}$$

Sustituyendo las expresiones anteriores se obtiene.

$$\eta_{il} = \frac{1}{\pi a^2}\frac{\left(\iint_{s'}\frac{\sqrt{D(\alpha)}}{R}\rho d\rho d\phi'\right)^2}{\iint_{s'}\left|\frac{\sqrt{D(\alpha)}}{R}\right|^2 \rho d\rho d\phi'} \qquad \text{(Ec. 5.3.45)}$$

Teniendo en cuenta la relación del ángulo β y la relación distancia focal a diámetro (Ec. 5.3.36) se llega a la expresión final para la eficiencia de iluminación.

$$\eta_{il} = 2\cot^2\frac{\beta}{2}\frac{\left(\int_0^{\beta}\sqrt{D(\alpha)}\tan\frac{\alpha}{2}d\alpha\right)^2}{\int_0^{\beta}D(\alpha)\sin\alpha d\alpha}$$

(Ec.5.3.46)

5.3.8 Eficiencia de Desbordamiento.

Se define como la relación entre la potencia radiada por el alimentador situada en el foco que llega al reflector y la potencia total radiada por el mismo.

$$\eta_s = \frac{W_{refl}}{W_{total}} = \frac{\int_0^{2\pi}\int_0^{\beta}P(\theta',\phi')ds}{\int_0^{2\pi}\int_0^{\pi}P(\theta',\phi')ds} = \frac{\int_0^{2\pi}\int_0^{\beta}D(\theta',\phi')d\Omega}{\int_0^{2\pi}\int_0^{\pi}D(\theta',\phi')d\Omega} = \frac{1}{4\pi}\int_0^{2\pi}\int_0^{\beta}D(\theta',\phi')d\Omega$$

(Ec. 5.3.47)

Dicha eficiencia depende el valor del ángulo β. En la Figura 5.3.14 se observa la eficiencia para una apertura elemental.

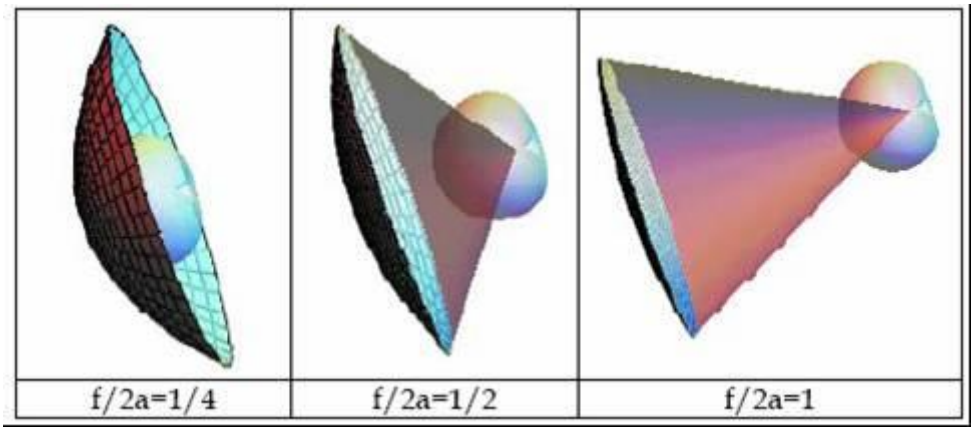


Figura 5.3.14 Eficiencia de Desbordamiento.

La eficiencia de desbordamiento es mayor para valores reducidos de la relación (f / 2 a). También depende de la antena situada en el foco. La eficiencia será mayor si la antena es mas directiva.

5.3.9 Eficiencia de bloqueo.

Se define como la relación entre las directividades o áreas efectivas de la antena con bloqueo o sin bloqueo.

En el caso de las aperturas circulares de radio "a" con distribución uniforme el área efectiva es:

$$A_{ef} = \pi a^2$$

(Ec. 5.3.48)

Si se bloque en el centro una zona de radio b, la nueva área efectiva será:

$$A_{ef}^{b} = \pi\left(a^2 - b^2\right) \qquad \text{(Ec. 5.3.49)}$$

La eficiencia de bloqueo será:

$$\eta_b = \frac{A_{ef}^{b}}{A_{ef}} = \frac{a^2 - b^2}{a^2} = 1 - \left(\frac{b}{a}\right)^2 = 1 - x^2 \qquad \text{(Ec. 5.3.50)}$$

5.3.10 Eficiencia de Polarizacion.

Las corrientes en el reflector tienen una orientación variable. El diagrama tiene componentes copular y Polarizacion cruzada. Se define eficiencia de Polarizacion como la relación entre la potencia radiada en la Polarizacion copolary la potencia radiada en las dos polarizaciones.

5.3.11 Eficiencia Total.

La eficiencia total incluye el efecto de perdidas por radiación fuera del reflector y de iluminación en la apertura, así como el resto de las perdidas por Polarizacion, bloqueo, etc.

$$\eta_t = \eta_s \eta_i = \frac{1}{2}\int_0^{\beta} D(\theta', \phi')\, d\Omega \left(2\cot^2\frac{\beta}{2}\right) \frac{\left(\int_0^{\beta} \sqrt{D(\alpha)}\tan\frac{\alpha}{2} d\alpha\right)^2}{\int_0^{\beta} D(\alpha)\sin\alpha d\alpha} \qquad \text{(Ec. 5.3.51)}$$

$$\eta_t = \cot^2\frac{\beta}{2}\left(\int_0^{\beta} \sqrt{D(\alpha)}\tan\frac{\alpha}{2} d\alpha\right) \qquad \text{(Ec. 5.3.52)}$$

6

VRML, "LENGUAJE PARA EL MODELADO DE LA REALIDAD VIRTUAL"

6.1 Introducción.

El lenguaje VRML [6] (Virtual Reality Modelling Language) permite la visualización y manejo de estructuras en tres dimensiones de forma muy sencilla y compacta, ya que el lenguaje de programación es un lenguaje de texto y se interpreta mediante un visor que se puede instalar como una extensión (*plug-in*) de los navegadores más habituales (Netscape e Internet Explorer).

La filosofía de VRML es muy similar a la de HTML, al ser un estándar abierto no está limitado a ninguna aplicación o plataforma. Los campos de aplicación del lenguaje VRML son muy amplias: Documentación técnica, arqueología, demostraciones, enseñanza, etc. En el campo que nos ocupa, el uso de VRML permite al alumno visualizar los diagramas de radiación de una antena.

6.2 Creación de archivos VRML.

Todo lo que se necesita para esto es un visor VRML y un editor del texto, como el notepad o el wordpad.

6.2.1 Descripción de un visor VRML.

La creación de gráficos es una labor asignada al visor VRML. Esta entidad sigue el modelo conceptual descrito en el estándar y representado esquemáticamente en la figura 6.1. Como puede apreciarse, el visor VRML es considerado una aplicación de representación que permite entradas del usuario en la forma de selección de archivos (explícita o implícitamente) y gestiones del interfaz gráfico (manipulación y navegación usando dispositivos de entrada). Las tres partes principales del visor son: *parser* o analizador, *world* o escena, y presentación audio/visual. El componente analizador lee el archivo VRML y crea la escena.

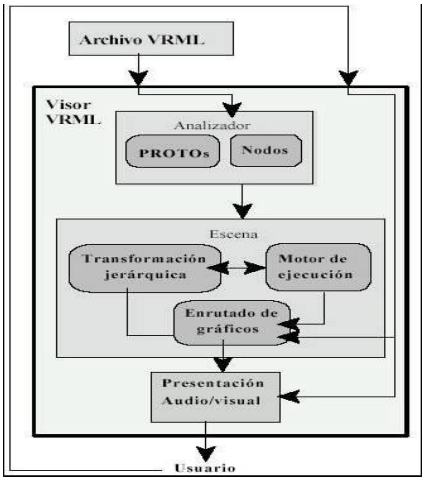


Fig. 6.1: Modelo conceptual del proceso de representación de entornos gráficos de VRML.

Ésta consiste en la transformación jerárquica y en enrutado de eventos. El componente de escena posee también un bloque dedicado al procesamiento de eventos, lectura y edición del grafo de rutas y realización de los cambios en la jerarquía de transformación, conocido como motor de ejecución.

Las entradas del usuario generalmente afectan a los sensores y a la navegación, de tal forma que se establece una realimentación desde la salida del visor hacia el usuario y la consecuente acción de éste.

6.2.2. Edición de un archivo VRML.

Para la edición del código de un archivo VRML se hace en un editor de texto, como el notepad o el wordpad y se guarda como *archivo.wrl.* La primera cosa que se debe saber es que todo archivo VRML comienza con una línea de **cabecera**, que es:

```
#VRML V2.0 utf8
```

Esto dice al navegador que está recibiendo un archivo de VRML, y la versión correspondiente del lenguaje. En este caso, es versión 2.0. VRML es sensible a mayúsculas y minúsculas, por lo que se debe asegurar de escribirlo tal y como esta aquí, incluyendo las mayúsculas y minúsculas. La parte del **utf8** dice al navegador qué standard de cadenas de texto debe emplear.

Cualquier línea que comience con una almohadilla (#) es un comentario, y será ignorado por el intérprete VRML, de modo que es lo que se usara para comentar los códigos. La primera línea es una excepción, y *es leída* por el navegador, a pesar de que tenga un # en el comienzo. Por lo tanto el siguiente código en VRML es válido:

```
#VRML V2.0 utf8
#Archivo VRML de ejemplo del tutorial de Floppy
```

El aspecto del visor se muestra en la figura 6.2, en la parte de la izquierda se ven una serie de botones, los tres de arriba indican el modo en que nos queremos desplazar (“walk”:andar, “fly” volar y “study” si lo que queremos es desplazar la escena quedándonos quietos), los cuatro siguientes definen cual movimiento se va a realizar(“plan”: acercarse o alejarse, “pan” desplazar la imagen, “turn” rotar en torno al eje y, “rol” rotar entorno al eje z), cualquier movimiento se realiza seleccionando sobre la zona de imagen y moviendo el ratón manteniendo el botón pulsado (el punto en el que se selecciona es el que se utiliza como referencia para girar, acercarse,.... por lo que hay que elegir correctamente entre los elementos de la escena).

De los botones de la parte inferior “goto” sirve para acercarse a un punto concreto (se marca sobre la imagen), “restore” permite volver a la situación inicial y “fit” encuadra toda la escena en el campo de visión. Precisamente, como las escenas generadas están en coordenadas objeto y, por defecto los visores “miran” a las coordenadas 0, 0,0 lo primero que hay que hacer es encuadrar la escena (también cuando debido a un movimiento brusco, saquemos la escena fuera del campo de visión).

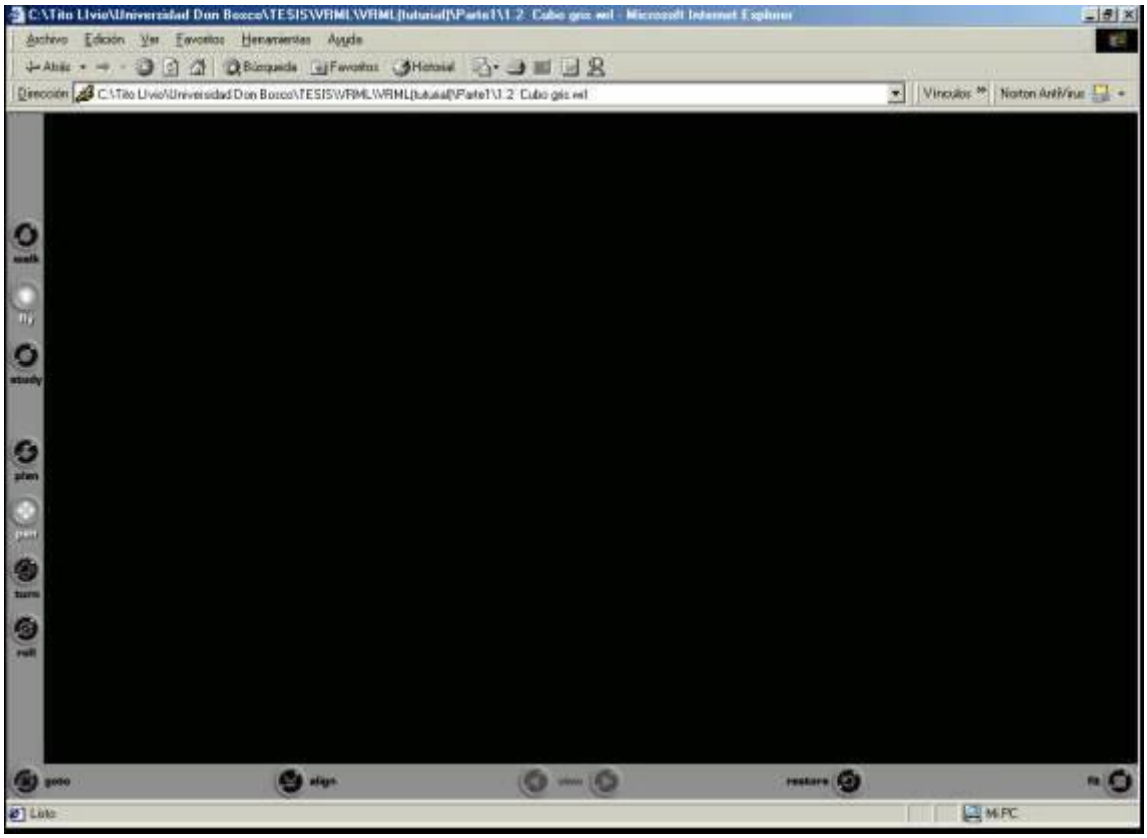


Figura 6.2. Aspecto del visor VRML.

6.2.3 Nodos y campos de un archivo VRML.

Un mundo VRML está hecho a partir de **nodos,** que son tipos de objeto. Dentro de estos nodos, se encontraran **campos,** que son propiedades del objeto. Los campos pueden ser cualquier cosa, desde el tamaño de una caja, a otro nodo dentro del primero. Los nodos se escriben como sigue:

```
Group {
    children [
    ]
}
```

El nodo **Group**, es el más simple, lo único que hace es agrupar otros nodos. El nodo se escribe en el archivo poniendo el nombre del nodo (Group), y después un par de llaves. Dentro de estas llaves tenemos los campos (children), que pueden contener otros nodos. De este modo los nodos pueden anidarse entre si, creando un tipo de jerarquía de nodos**.**

Los nombres de los nodos comienzan con *mayúscula* (Group, Transform, IndexedFaceSet), mientras que los de los campos lo hacen con *minúscula*. Aún así, pueden tener una letra central mayúscula si el nombre es compuesto (children, translation, coordIndex).

6.3 Tipos de valores de los campos.

Los archivos VRML consisten en grupos de nodos. Estos nodos también pueden contener otros nodos, y también puede contener campos.
Cada campo contiene un tipo que gobierna qué datos puede contener, y cuántos de él. Existen varios tipos de campos en VRML los cuales son:

- **SFBool**
  Este es un valor booleano simple que puede tomar el valor "TRUE" o "FALSE".

- **SFColor & MFColor**
  Este campo contiene un color simple conformado por tres números de punto flotante entre 0 y 1 que corresponden a los valores rojos, verdes y azules de ese color, por ejemplo 0 1 0 es verde. MFColor es un campo que contiene colores múltiples, por ejemplo [0 1 0, 1 0 0, 0 0 1].

- **SFFloat & MFFloat**
  SFFloat es un valor simple de punto flotante, por ejemplo 7.5. MFFloat son varios valores de punto flotante, por ejemplo [1.0, 3.4, 76.54].
- **SFImage**
  SFImage es una imagen bidimensional, ya sea en color o gris. Consiste de:
  Dos enteros, representando la anchura y altura de la imagen.
  - Un entero que representa el número de componentes en la imagen. 1 es sólo los niveles grises, 2 son grises con transparencia, 3 son el color en RGB, y 4 son RGB con transparencia.
  - Después de éstos, hay *anchura* * *altura* en hexadecimales, consistiendo en 2 dígitos para cada componente. Así que, 0xFF sería blanco en una imagen de un componente, y 0xFF00007F sería rojo medio transparente en una imagen del 4 componente.

  Los píxeles se especifican de izquierda a derecha y de abajo hacia arriba.

- **SFInt32 & MFInt32**
  Uno solo o una lista de números enteros de 32 bits. Éstos pueden estar en formato decimal o hexadecimal. Los números en hexadecimal comienzan con el 0x, por ejemplo el 0xFF es 255 en decimal.

- **SFNode & MFNode**
  SFNode es un solo nodo, y MFNode es una lista de nodos. Nota que los campos children en muchos nodos son del tipo MFNode.

- **SFRotation & MFRotation**
  Estos campos especifican una rotación sobre un eje. Esto es realizado por cuatro números del punto flotante. Los primeros tres especifican las coordenadas X Y Z para el vector correspondiente al eje sobre el cual rotar, y el cuarto es el número (ángulo) en radianes para rotar. SFRotation es un solo set de valores, MFRotation es una lista.

- **SFString & MFString**
  Este tipo contiene una lista de caracteres en el set de caracteres utf-8. ASCII es un subconjunto de utf-8, de modo que no hay que preocuparse sobre caracteres diferentes. Un string (SFString) se especifica como "Hola", entre comillas. Una lista (MFString) se parecería a: ["Hola", "Mundo"].

- **SFTime & MFTime**
  Uno solo o una lista de valores de tiempo. Estos se especifican como números de punto flotante que representan el número de segundos transcurridos desde que pasó la medianoche del 1 de enero1970. Esto tendrá más sentido cuando cubramos los eventos.
- **SFVec2f & MFVec2f**
  Uno solo o una lista de vectores 2D. Un vector 2D es un par de números de punto flotante.

- **SFVec3f & MFVec3f**
  Uno solo o una lista de vectores 3D. Un vector 3D es un triple de números de punto flotante.

6.3.1 Eventos

Así como los campos, la mayoría de los nodos contienen *eventos.* Hay dos tipos de eventos:

- ✓ **eventOut**s: son eventos emergentes (salientes) que generan información como el cambiar un valor o el tiempo de un clic del ratón.
- ✓ **eventIn**s: son eventos entrantes que aceptan información de fuera del nodo y hacen algo con él. Cada evento tiene un tipo de dato asociado a él.

Algunos nodos tienen campos que son *exposed (expuestos).* Esto significa que el nodo tiene dos eventos definidos para ese campo, **set_nombredecampo** y **nombredecampo_changed.**
Éstos son un **eventIn** y **eventOut** para el campo que puede usarse para cambiar su valor y notificar al mundo externo cuando ha cambiado. Si se utiliza **set_nombredecampo** para cambiar el valor del campo del evento, el nodo generará un **nombredecampo_changed.** Para una facilidad de uso, el **set_** y **_changed** de los eventos puede omitirse, y el navegador trabajará con el evento que está usándose.
Si un campo no es expuesto, no puede intercambiar eventos, y el valor en el archivo será el usado en todo momento.

6.3.2 Routes

Para hacer cosas útiles con los eventos, nosotros necesitamos unirlos de algún modo. Esta unión es conocida como ROUTE. Por ejemplo, para dirigir un touchTime eventOut a un startTime eventIn (por ejemplo para reproducir un sonido con un clic del ratón), nosotros dirigiríamos el evento como sigue:

ROUTE SENSOR.touchTime TO SOUND.startTime

Así que ésta parte de código dirigirá el evento del touchTime de un TouchSensor (que cubriremos luego) a un evento startTime en un nodo de sonido (que también cubriremos luego). Por consiguiente, cuando el TouchSensor es pulsado con el ratón, el sonido se activará. Se necesitara usar DEF para cada nodo que se precise dirigir hacia o desde, para que tengan nombres individuales. De esta manera el TouchSensor y el nodo del sonido se deberían definir con un nombre.

```
DEF SENSOR TouchSensor {
}

DEF SOUND Sound {
}
```

Excepto por el hecho de que deberían contener campos, obviamente. Si se tiene varios objetos con el mismo nombre (usando el USE), y se enruta hacia o desde ellos, todos los objetos serán afectados.

6.4 Creación de mundos VRML

6.4.1 Nodo WorldInfo.

Este nodo contiene la información general sobre el mundo, es como un título para el mundo. Este se despliega en la barra del título de la ventana del navegador, igual que la etiqueta <TITLE> en HTML. WorldInfo también puede contener más información sobre el archivo. Se puede escribir aquí cualquier cadena de texto o cualquier número. Un ejemplo del nodo WorldInfo se observa debajo:

```
WorldInfo {
  title "Floppy's VRML97 Tutorial Example 1"
  info ["(C) Copyright 1999 Vapour Technology"
       "vrmlguide@vapourtech.com"]
}
```

Se pueden tener múltiples cadenas de texto en el campo info, siempre colocándolos entre corchetes.

6.4.2 Las Formas básicas

Colocaremos un objeto Box, con los valores predefinidos. Hay un par de nodos requeridos como el Appearance y otros, que se definirán mas adelante. Hay que recordar, si no se especifica un valor para un campo, se usará el valor por defecto. Así que éste usará los valores predefinidos. Ahora agregaremos un nodo Shape al mundo, como se muestra el código abajo.

```
#VRML V2.0 utf8

WorldInfo {
  title "Floppy's VRML97 Tutorial Example 1"
  info ["(C) Copyright 1999 Vapour Technology"
       "vrmlguide@vapourtech.com"]
}
```

```
Shape {
  appearance Appearance {
    material Material {
    }
  }
  geometry Box {
  }
}
```

Dentro del nodo Shape existen dos campos para describir al objeto. Éstos son appearance y geometry que describen la apariencia y forma del objeto respectivamente. En la figura 6.3 se muestra visor VRML de lo descrito.

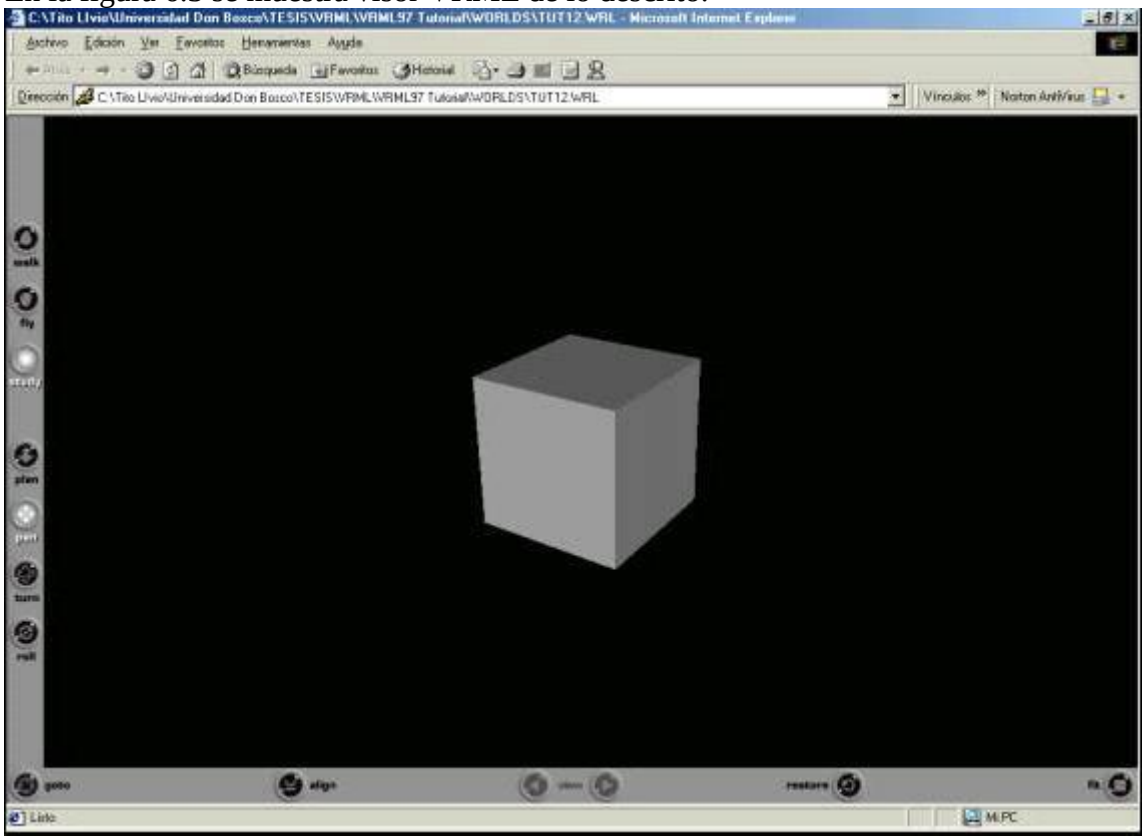


Figura 6.3. Figura del cubo.

6.4.2.1 Reutilizar un Objeto

Si se tiene muchos objetos idénticos, seria innecesario escribir muchas veces objetos del mismo tipo exacto. Por consiguiente, se puede reusar las definiciones anteriores. Usando nuestra caja, podríamos definirla con el nombre FBOX. Entonces, siempre que queramos usar esa caja de nuevo, simplemente podemos teclear USE FBOX en lugar de la definición completa. El código se muestra abajo.

```
DEF FBOX Shape {
  appearance Appearance {
    material Material {
    }
  }
  geometry Box {
  }
}
USE FBOX
```

La cabecera VRML y el nodo WorlInfo no se han incluido en el código anterior por cuestiones de ahorro de espacio, pero se da por sabido que en el editor de texto si ira incluido, al igual que en códigos que se muestren a lo largo del texto.

6.5 Traslaciones, rotaciones, y escalas

Todas las distancias en VRML se miden en metros. El sistema de coordenadas en VRML funciona tal y como se muestra abajo:

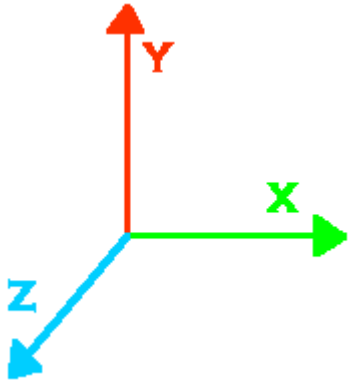


Figura 6.4. Sistema de coordenadas en VRML.

El eje X es horizontal, Y es vertical y Z atraviesa la pantalla hacia fuera como se ve en el diagrama de arriba.

Las rotaciones en VRML funcionan con la regla de la mano derecha, es decir igual que el sentido de las agujas del reloj en una dirección positiva. Esto es para las rotaciones sobre cualquier eje, entonces si se quiere rotar un objeto 90 grados alrededor del eje X, se tendría que utilizar en 90 grados la rotación negativa. Esto también aplica a las rotaciones sobre los ejes arbitrarios. Otro punto para resaltar es que todas las rotaciones son medidas en radianes y no en grados

6.5.1 Las transformaciones

Para hacer cualquier uso de nuestro mundo, se necesita poder *transformar los* objetos. VRML tiene tres tipos de transformaciones que nosotros podemos aplicar a los objetos. Éstas son traslaciones**,** rotaciones, y escalas. Éstas son usadas en el nodo Transform. Se puede tener solamente una rotación, por ejemplo, las transformaciones dentro de Transform se aplican a los children del nodo. Esto se llama *anidación (nesting)*, dónde un nodo puede tener cualquier número de nodos dentro del children. La sintaxis para esto se muestra en el ejemplo de abajo, junto con la sintaxis del nodo Transform.

```
Transform {
  translation 1 1 1
  rotation 0 1 0 0.78
  scale 2 1 2
  children [
    USE FBOX
  ]
}
```

El nodo Transform puede tener otro anidado dentro de él como un children, lo que permite hacer secuencias de transformaciones. Hay que recordar el orden de las transformaciones. Una rotación seguida por una traslación no es igual que una traslación

seguida por una rotación. Dentro del nodo Transform**,** las transformaciones se llevan a cabo en el orden estricto: Escala, entonces Rota, luego Traslada. Así, si se quiere una traslación seguida por una rotación, se necesita anidar un nodo Transform dentro de otro.

6.5.2 Traslación y Escala

Estas transformaciones son muy similares. Ambas toman tres argumentos; valores de x, y, z. La traslación mueve el centro del objeto esas distancias en la apropiada dirección. La escala multiplica el tamaño del objeto por estos valores en las direcciones apropiadas. Una traslación de 0 en una dirección dejará el objeto sin afectar en esa dirección. Un factor de escala 0 hará el objeto infinitamente delgado en esa dirección, cosa que no es normalmente deseable. Un factor de escala 1 no produce ningún efecto.

Es muy importante recordar que el escalado toma el lugar relativo del *origen*, no el centro del objeto. Así para que para escalar desde el centro de un objeto, se debe asegurar que el objeto se centra en el origen. Esto explica por qué el escalado se hace primero, antes que cualquier rotación o traslación.

6.5.3 La Rotación

La rotación es ligeramente diferente de los dos tipos anteriores. Toma cuatro argumentos. El primero son las tres coordenadas que definen el eje de rotación y el último es el ángulo para rotar, en radianes. Así, para rotar 1 radián sobre el eje de Y, por ejemplo, se debe escribir:

```
Transform {
  rotation 0 1 0 1
  children [
    USE FBOX
  ]
}
```

La longitud del eje de la rotación puede ser cualquier valor, no necesariamente 1. El eje de rotación uno lo elige, se puede rotar sobre cualquier eje que se requiera. Por ejemplo, un eje de 1 0.3 2.45 es absolutamente válido. Sin embargo, puede ser difícil de visualizar, sobre todo si no tienes práctica.

6.6 Los nodos de la apariencia.

Como se ha visto con el objeto FBOX que definimos antes, el nodo Shape tiene un campo dentro llamado appearance**.** Este se usa para contener un nodo Appearance, como se muestra abajo:

```
DEF FBOX Shape {
  appearance Appearance
        {  material Material { }
        }
  geometry Box { }
}
```

El código anterior tiene un campo dentro del nodo Appareance, el campo material. Esta manera de tener nodos dentro de nodos puede parecer complicado, pero permite definir las apariencias globalmente usando **DEF** y **USE**, esto es útil para tener muchos objetos

con la misma apariencia. El nodo Appearance puede contener el campo material o el campo texture. El campo material contiene un nodo Material. Un campo texture contiene uno de varios tipos de nodos de texturas.

6.6.1 El campo material

El campo material contiene un nodo Material y este puede contener cualquiera los siguientes seis campos:

- **diffuseColor**
  El color normal del objeto.
- **specularColor**
  El color resaltado en los objetos brillantes.
- **emissiveColor**
  El objeto irradia una luz propia de ese color. Pero no iluminará ningún otro objeto con esa luz.
- **ambientIntensity**
  La cantidad de luz del ambiente que el objeto refleja.
- **shininess**
  Cuan reflexivo es el objeto.
- **transparency**
  Cuan transparente es el objeto. Algunos navegadores no soportarán objetos parcialmente transparentes.

Los primeros tres de estos campos poseen valores de color (RGB), y los últimos tres son un número simple entre 0 y 1. Los colores son especificados con los componentes verde, rojo y azul (RGB). Se puede hacer cualquier color, combinando los valores (RGB).
Si queremos un cubo verde semitransparente brillando, definimos el código de la siguiente forma:

```
Shape
{ appearance Appearance
            { material Material
                    {  emissiveColor 0 0.8 0
                       transparency 0.5
                    }
            }
  geometry Box {}
}
```

Y el ejemplo del código anterior lo podemos ver en la figura 6.5.

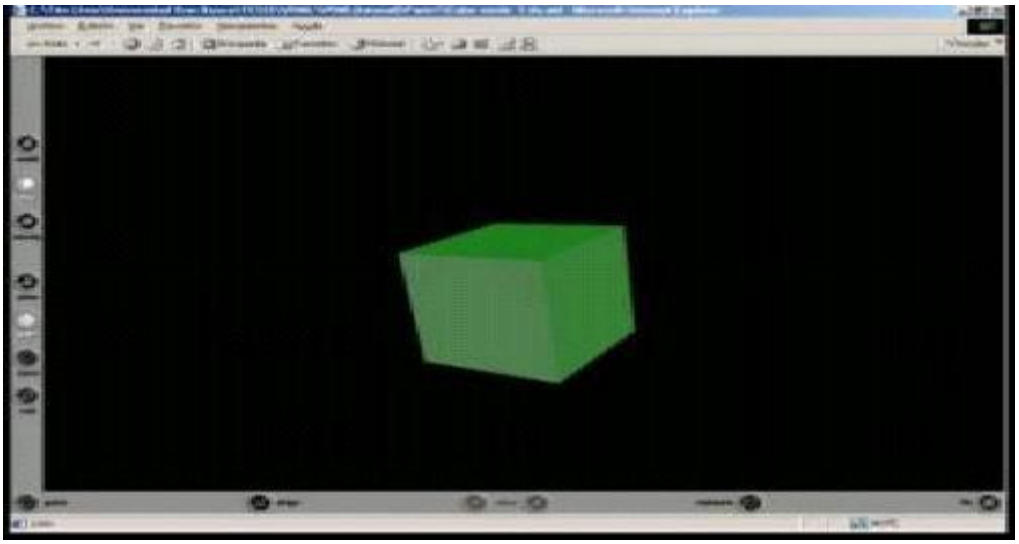


Figura 6.5. Figura del cubo semiverde.

6.6.2 El campo texture

Aparte de los colores para que todo esto luzca *mucho* mejor, se puede mapear una textura a nuestros objetos. Esto se logra utilizando el campo **texture** del nodo Appearance**.** Este campo contiene uno de tres tipos de nodo texture.

6.6.2.1 El nodo ImageTexture.

Éste es un mapa de textura básico, que traza una imagen inmóvil sobre un objeto. El nodo puede trazar una textura sobre el objeto con formato de archivo JPEG o PNG, el archivo GIF no es valido.
El nodo contiene tres campos. El primero, el url especifica la imagen para usar en un formato de URL normal. Se pueden especificar una lista de imágenes dentro de corchetes y el navegador mostrara la primera que encuentre. Los otros dos son repeatS y repeatT que controlan si la textura se repite en la dirección horizontal (S) o en la vertical (T). Estos toman valores boléanos de TRUE (Verdadero) o FALSE (Falso). Sólo son realmente muy útiles cuando se combinan con un TextureTransform.
Se puede especificar la información de transparencia en las imágenes en cuyo caso reemplazará a la transparencia del objeto original. Si se usa una textura en escala de grises, el diffuseColor se multiplicara por la intensidad de la textura hasta crear la textura actual. De hecho, se pueden crear muchos efectos combinando el nodo Material e ImageTexture.

6.6.2.2 El nodo MovieTexture.

MovieTexture toma una película en formato MPEG y la mapea en un objeto de la misma manera que con ImageTexture. Este tiene los mismos tres campos, pero posee los siguientes:

- **speed**

Un valor de velocidad 1 significa la velocidad predefinida, 2 es dos veces más rápido. Un valor de 0 siempre desplegará el primer fotograma.

- **loop**
  Un valor booleano (TRUE o FALSE), especificando si la película se repite o no.
- **startTime**
  Cuándo comienza la película, medido en segundos desde la medianoche del 1 Ene 1970.
- **stopTime**
  Cuándo se detiene la película, medido en segundos.

6.6.2.3 El nodo PixelTexture

Este nodo permite definir tus propias texturas a mano en el archivo VRML. Esto parece ineficaz, pero lo que hace tiene sus usos. Este tiene un campo image en vez del URL.
El campo image consiste en dos números especificando el ancho y la altura de la textura, seguidos por otro número que aporta el número de los componentes. Los colores con un componente son los de la escala de grices, los colores de dos componentes son los grises con transparencia, tres son el color en RGB, y cuatro son RGB con transparencia.

Después de estos argumentos, sigue una lista de píxeles que son los números en hexadecimal con un byte por componente, así un píxel de 4 componentes que es rojo y 50% transparente sería 0xFF00007F. Los píxeles se ordenan de abajo-izquierda hacia arriba-derecha.

Un ejemplo sobre este nodo se observa en la figura 6.6, y el código se muestra a continuación:

```
Shape
 {appearance Appearance
        {texture PixelTexture
              {image 2 2 3 0xFF0000 0x00FF00 0x0000FF 0xFF0000
              }
        }

  geometry Cylinder {}
 }
```

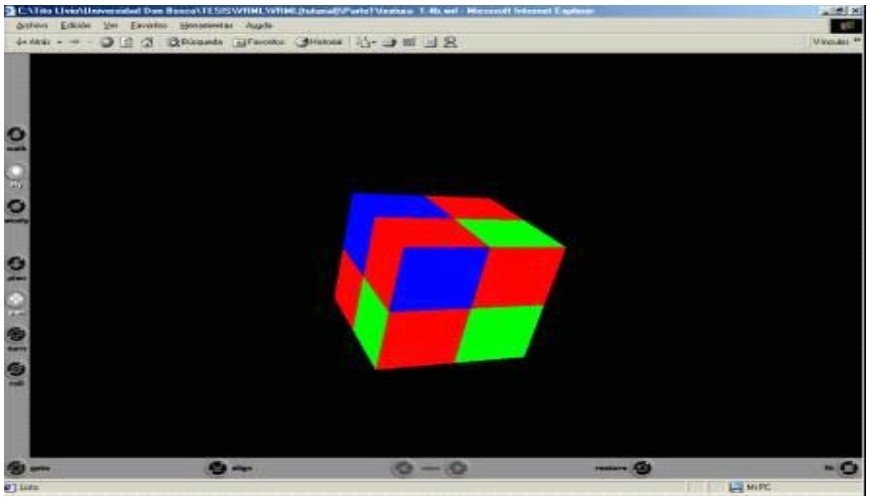


Figura 6.6. Cubo con PixelTexture.

En resumen de esta sección, tenemos dos cajas, una verde transparente, y otra con una textura de ladrillo en el siguiente código. Hay que observar, que no podemos más usar DEF en el nodo Shape, ya que las cajas tienen texturas diferentes, el código se muestra a continuación.

```
Shape {
  appearance Appearance {
    material Material {
      emissiveColor 0 0.8 0
      transparency 0.5
    }
  }
  geometry Box {
  }
}

Transform {
  scale 2 0.5 2
  rotation 0 1 0 0.78
  translation 0 -1.5 0
  children [
    Shape {
      appearance Appearance {
        texture ImageTexture {
          url "brick.jpg"
        }
      }
      geometry Box {}
    }
  ]
}
```

El resultado del código anterior se muestra en la figura 6.7.

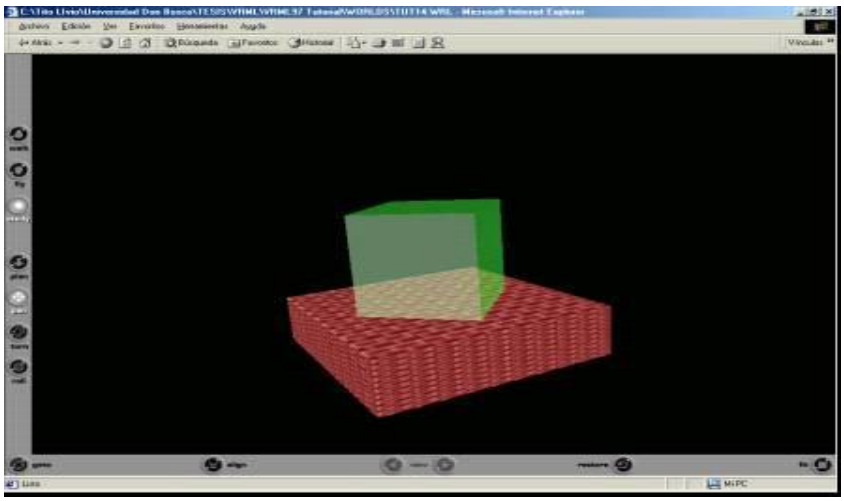


Figura 6.7. Caja verde y caja con textura de ladrillo.

6.7 Nodos de figuras geométricas

6.7.1 La caja.

Es un cubo o paralelepípedo, el Box predefinido es un cubo, de dos metros por lado, centrado en el origen. Para realizar un Box de un tamaño diferente, puede definirse con un tamaño explícito:

```
geometry Box
    { size 5.5 3.75 1.0
    }
```

Esto crea un Box con X, Y y dimensiones de Z como fueron especificadas en el parámetro size.

6.7.2 El cilindro

El nodo Cylinder es ligeramente más complejo. El valor por defecto es un cilindro de +1 a -1 en todas las dimensiones, (radius 1, height 2). También queda centrada en el origen. Para especificar un tamaño diferente, nosotros podemos usar los campos radius (radio) y height **(**altura**).**
Hay también otros tres campos, side**,** top, y bottom, estos son valores Booleanos (TRUE o FALSE), y dicen al navegador si debe mostrar la sección apropiada del cilindro. Éstos valores predefinidos están en TRUE, gracias a lo cual se necesita especificarlos mayoría de las veces. Sin embargo, si tuviéramos un cilindro dónde un extremo este oculto por otro objeto, merecería la pena quitar ese extremo, ya que con esto se reduciría gran cantidad de trabajo del navegador, acelerando la ejecución de un mundo.

```
geometry Cylinder
    {radius 0.5
     height 10
     top FALSE
    }
```

Esto da un cilindro alto, delgado sin la cara superior (top).

6.7.3 El cono

Muy parecido al cilindro, sólo que en lugar del campo radius, tenemos un bottomRadius que hace exactamente lo que sugiere: especifica el radio de la base. No hay tampoco ningún campo top, ya que los conos no tienden a tener cimas.

```
geometry Cone
    {bottomRadius 5
     height 10
     side TRUE     #Esto no es realmente necesario, ya que está por defecto.
     bottom FALSE
     }
```

El centro está en el medio, a medio camino del cono. Ahora, cuando se considera la caja que contiene el cono parece obvio, pero no es totalmente intuitivo. Así que, un cono predefinido se extenderá 1 metro sobre el plano Y y 1 metro debajo de este plano, teniendo un radio de 1 en la base.

6.7.4 La esfera

El nodo Sphere tiene sólo un campo que es su radius(radio)**.**

```
geometry Sphere
      {radius 10
      }
```

Las texturas son aplicadas envolviéndose alrededor de la esfera estrechándose en los polos. Si tuvieras una textura con una cuadrícula en ella, las líneas verticales se acercarían cada vez mas cuando estén próximas a los dos polos, mientras que las líneas horizontales quedarían con las mismas distancias todo el tiempo.

6.7.5 El texto & FontStyle

Este nodo crea textos 2D en el mundo virtual. Realmente es muy simple. El nodo Text tiene sólo cuatro campos. El primero es string (serie de caracteres manipulados como un grupo) dónde se define una cadena de caracteres o una lista de cadenas a desplegar. El campo fontStyle contiene un nodo FontStyle. Los últimos dos campos son maxExtent dónde se especifica la anchura máxima (en metros) del texto, y length que es una lista de longitudes (de nuevo en metros) para cada string, para que se pueda especificar una anchura específica para cada uno de ellos. Si se especifican las length, el navegador modificara el tamaño del texto para ajustarlas a ese tamaño.

```
geometry Text
      {string ["Hola", "Mundo"]
        fontStyle USE HELLOFONT
        maxExtent 5
        length [3, 3]
      }
```

6.7.6 Nodo FontStyle

Este es más complejo, la mejor manera de empezar aquí es mostrar todos los campos.

```
FontStyle
    {size
     family
     style
     horizontal
     leftToRight
     topToBottom
     language
     justify
    spacing
    }
```

El size es la altura, en metros, de la línea de texto. El campo family puede tomar uno de tres valores, y altera la tipografia de los caracteres. Los tres tipos son "SERIF", "SANS", o "TYPEWRITER". Para cambiar la apariencia del texto, se puedes usar el campo style. Puede tomar cualquiera de estos valores: "PLAIN", "BOLD", "ITALIC", o "BOLD ITALIC". Horizontal es un valor booleano que muestra si el texto es horizontal ("TRUE") o vertical ("FALSE"). leftToRight (Izquierda a derecha) y topToBottom (De arriba a abajo) también son booleanos, y su operación es un poco obvia. Con language, el sistema de caracteres utf-8 puede aparecer de manera distinta, depende en que idioma se lea. justify es muy útil, y puede ser cualquiera de estos "BEGIN", "MIDDLE", o

"END." spacing es la cantidad de espacio entre las líneas de texto. 1 es normal, y 2 corresponden a doble-espacio (una línea en blanco entre cada línea).

Resumen.

Todos los nodos de la geometría vistos hasta ahora (y todos los otros que encontraremos después) se centran en el origen del sistema de coordenadas local. Por ahora, sólo hay que recordar que el punto para transformar la posición de los objetos y todo lo demás es el centro del objeto.

A continuación se muestra minimonumento en la figura 6.8, en base a lo cubierto en esta sección y su respectivo código.

```
Shape {
  appearance Appearance {
    material Material {
      diffuseColor 0 0.5 0
      emissiveColor 0 0.8 0
      transparency 0.5
    }
  }
  geometry Cylinder {
    radius 0.35
    height 3
    bottom FALSE
  }
}

Transform {
  translation 0 -2 0
  children [
    Shape {
      appearance Appearance {
        texture ImageTexture {
          url "brick.jpg"
          repeatS TRUE
          repeatT TRUE
        }
      }
      geometry Box {
        size 4 1 4
      }
    }
  ]
}

Transform {
  translation 1.6 -1 1.6
  children [
    Shape {
      appearance Appearance {
        material Material {
          diffuseColor 1 0 0
        }
      }
      geometry DEF ICKLECONE Cone {
        bottomRadius 0.3
        height 1
        bottom FALSE
      }
    }
  ]
}

Transform {
  translation -1.6 -1 -1.6
  children [
    Shape {
      appearance Appearance {
        material Material {
          diffuseColor 1 0 0
        }
      }
      geometry USE ICKLECONE
    }
  ]
}

Transform {
  translation -1.6 -1 1.6
  children [
    Shape {
      appearance Appearance {
        material Material {
          diffuseColor 1 0 0
        }
      }
      geometry USE ICKLECONE
    }
  ]
}

Transform {
  translation 1.6 -1 -1.6
  children [
    Shape {
      appearance Appearance {
        material Material {
          diffuseColor 1 0 0
        }
      }
      geometry USE ICKLECONE
    }
  ]
}

Transform {
  translation 0 2.25 0
  children [
    Shape {
      appearance Appearance {
```

```
        texture ImageTexture {
          url "me.jpg"
        }
      }
      geometry Sphere {
        radius 0.75
      }
    }
  ]
}

Transform {
  translation 0 -2.2 2.01
  children [
    Shape {
      appearance Appearance {
        material Material {
          diffuseColor 0 1 0
        }
      }
      geometry Text {
        string "FloppyWorld"
        fontStyle FontStyle {
          size 0.8
          justify "MIDDLE"
        }
      }
    }
  ]
}
```

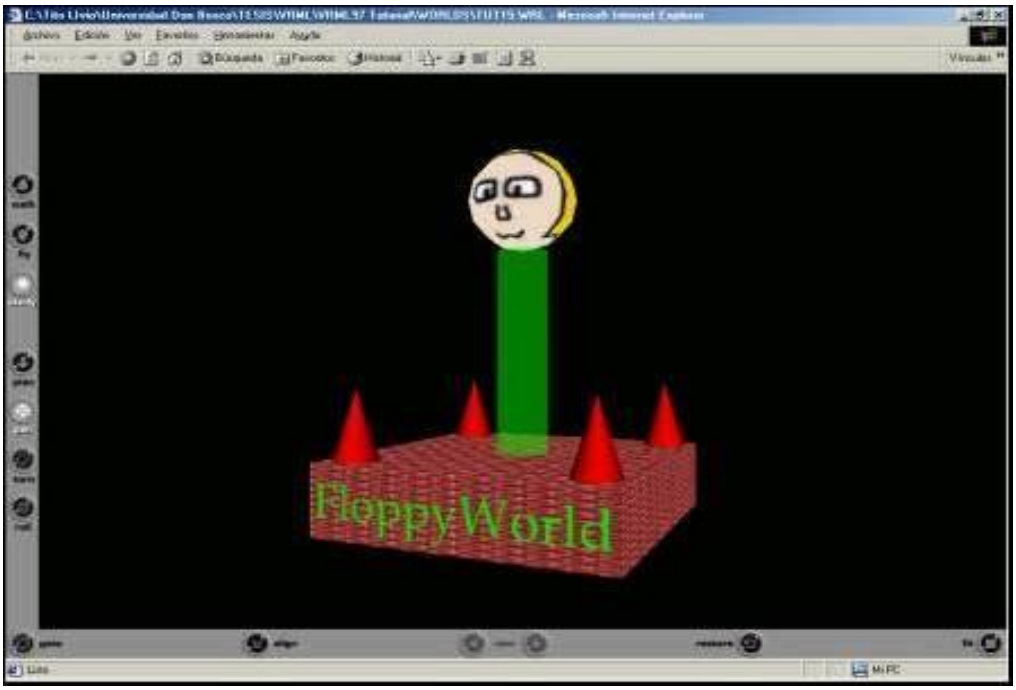


Figura 6.8. Mini monumento.

6.8 Cámaras en mundos 3D

Los navegadores de VRML crean un punto de entrada predefinido para una escena que normalmente está a lo largo del eje +Z a la distancia necesaria para que todo el mundo se despliegue en la ventana. Sin embargo, podria no querer empezar desde esta posición, o se podría querer que el espectador pudiera seleccionar uno de varios puntos de vista. Para hacer esto, se debe poner cámaras extras en la escena usando el nodo Viewpoint.

El nodoViewpoint tiene varios campos. El primero es el campo position que es un SFVec3f (es decir, un triple valor de punto flotante que representan la posición de X, Y y Z). Esto define la posición de la cámara en el mundo. El próximo es el campo orientation que es un SFRotation. Este tiene tres números de punto flotante que definen la rotación de la cámara, de la misma manera que una rotación normal. La orientation predefinida está mirando en la dirección de -Z con +X al derecho y +Y encima.

El próximo campo es fieldOfView. Este es un número de punto flotante en radianes entre 0 y pi, definiendo el ángulo del campo visual. Esto se muestra en la figura 6.9. El valor por defecto es 0.78 radián, que da un campo de vista normal.

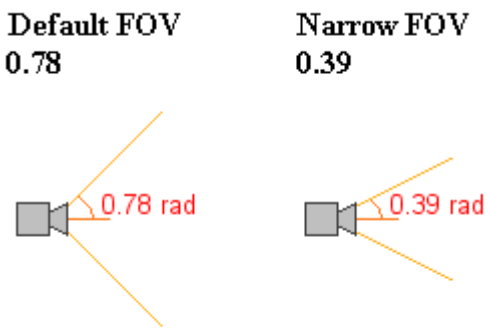


Figura 6.9. Descripción del campo fieldOfView.

El nodo Viewpoint también tiene un campo description que el navegador puede usar para desplegar una descripción de la cámara. Es un campo SFString**.** La sintaxis completa para el nodoViewpoint es:

```
Viewpoint
      {position 0 5 5
       orientation 1 0 0 0.39
       fieldOfView 0.39
       description "Camara 1"
       }
```

Esto define una cámara en 0 5 5 mirando hacia abajo ligeramente a 22.5 grados, con un FOV estrecho, y una descripción "Cámera 1"
Se puede usar diferentes cámaras para crear entradas diferentes a la escena. Especificando el nombre de la cámara en un vínculo de la página, esa cámara se utiliza en la entrada a la escena. Por ejemplo, para entrar en una escena en una cámara llamada "CAM1" se podría vincularla así:

```
world.wrl#CAM1
```

Esto significa que la cámara ha sido previamente definida para llamarse CAM1. La descripción se usa en el navegador, pero este método usa el nombre especificado en DEF.

6.9 El fondo.

El fondo predefinido es negro. Hay dos maneras en que podemos cambiar el fondo de nuestra escena. Una es especificar el color para el fondo, y la otra es proporcionar imágenes a ser proyectadas en el fondo. Estas dos se llevan a cabo con el nodo Background.
El nodo Background tiene varios campos usados para el color del fondo. Estos son:

```
Background {
  MFFloat    groundAngle      []
  MFColor    groundColor      []
  MFFloat    skyAngle         []
  MFColor    skyColor         [0 0 0]
}
```

Los valores predefinidos se muestran anteriormente. El valor por defecto es un solo skyColor de 0 0 0, negro. El skyColor es una lista de valores SFColor, correspondiendo a la secuencia de colores a ser desplegados en el fondo, desde el punto mas arriba, hacia abajo. Así que el primer valor se muestra arriba, y el próximo se despliega debajo de él y así sucesivamente. El skyAngle es el ángulo en que cada banda de color se va a mostrar. El primer color se despliega automáticamente a un ángulo de 0 (recto), así que no precisas incluir ese. Habrá siempre, por consiguiente, un ángulo menos que colores. El navegador interpola entre los colores en el fondo para que se consiga un efecto mezclado suave. Por ejemplo, el código siguiente daría una mezcla entre rojo (recto arriba), verde (a 45 grados), y azul (recto delante).

```
Background {
  skyColor [1 0 0, 0 1 0, 0 0 1]
  skyAngle [0.78, 1.57]
}
```

GroundColor y groundAngle tienen el mismo efecto que arriba, pero un ángulo de 0 corresponde a la recta debajo, en lugar de la recta de arriba en el cielo. Para tener un solo color como fondo, simplemente se especifica un solo skyColor. Para conseguir un horizonte afilado, se necesitas usar groundColor y skyColor, dado que si sólo se usa uno, interpolará suavemente entre los colores.

6.9.1 Los panoramas

Hay otra manera de proveer un fondo a nuestros mundos. Esto es proporcionar varias texturas a ser trazadas en un cubo alrededor del mundo. Este cubo siempre esta centrado en el espectador, entonces nunca podrá acercarse a este cubo, sin importar cuan lejos se mueva. Esto se hace con otros seis campos en el nodo Background**.** Éstos especifican seis urls de imágenes a ser trazadas en el cubo, tal y como se muestra abajo.

```
Background {
  MFString   backUrl      []
  MFString   bottomUrl    []
  MFString   frontUrl     []
  MFString   leftUrl      []
  MFString   rightUrl     []
  MFString   topUrl       []
}
```

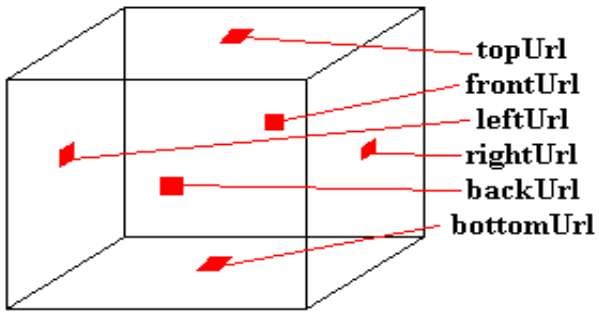


Figura 6.10. Descripción del nodo Background.

Si se especifica imágenes con transparencia, el color de fondo especificado previamente en los campos descritos se mostrará a través suyo. Normalmente, si no se tiene transparencias, no se debería especificar el color del fondo, ya que él el navegador bajará su rendimiento.

6.9.2 La niebla.

Si se quiere un efecto de niebla, VRML puede hacer eso también. Sólo se tiene que incluir un nodo Fog en el archivo. Los objetos en la distancia serán disimulados entonces por la niebla, y aparecerán cuando se acerque a ellos. La sintaxis de un nodo de Fog es como sigue:

```
Fog {
  SFColor    color           1 1 1
  SFString   fogType         "LINEAR"
  SFFloat    visibilityRange   0
}
```

El campo color especifica el color, entonces se tiene niebla, smog, humo, una llovizna roja o algo igualmente raro. El type es el tipo de niebla que gobierna cuan rápidamente los objetos se desvanecen en la niebla. Hay dos opciones aquí: "LINEAR" y "EXPONENTIAL." "LINEAR" da un desvanecimiento lineal en la niebla, y "EXPONENTIAL" da el ocultamiento más exponencial, dando una apariencia de niebla más natural. El último campo es visibilityRange. Más allá de esta distancia del espectador, nada es visible. Dentro de este rango, los objetos son disimulados parcialmente por la niebla. Un visibilityRange de 0 significa que la niebla no afecta nada, cualquiera sea su distancia.
El nodo Fog es útil para la optimización, así como para efectos de fantasía. Si se tiene un nodo Fog y un nodo Background, el fondo no es afectado por la niebla, así que se muestra a través normalmente.

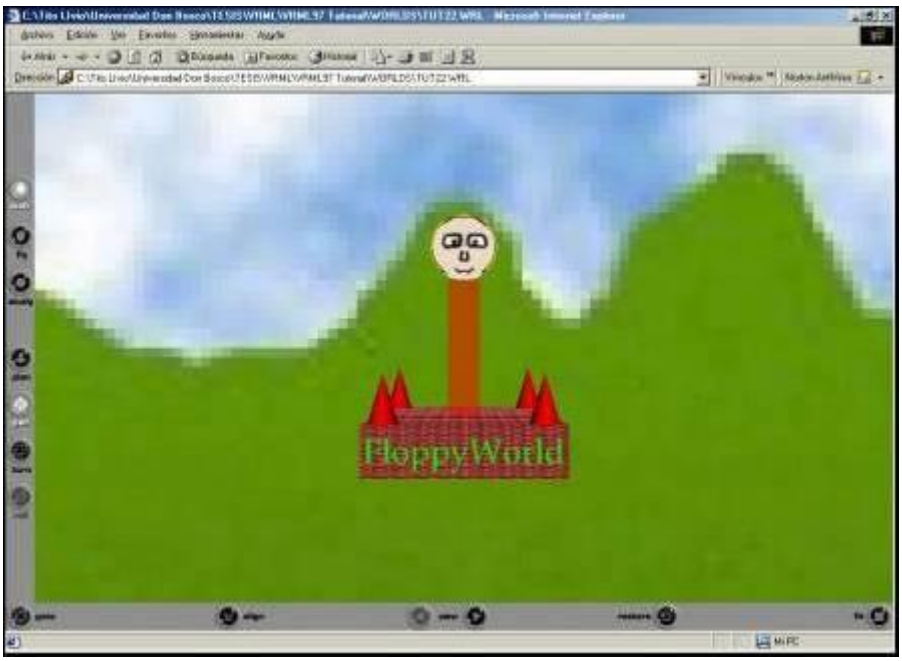


Figura 6.11. Ejemplo utilizando el nodo Background.

6.10 El sonido

Para hacer el mundo más interesante, se le puede agregar sonido. Estos pueden ser sonidos de ambiente, timbres, sirenas, lo que mejor a uno le parezca. Todo esto se hace con dos tipos de nodos, Sound y AudioClip.

6.10.1 El nodo Sound

Se usa para especificar la posición y orientación de una fuente de sonido en un mundo. El nodo Sound se coloca en un cierto lugar, y se podrá oír el sonido dependiendo del lugar en que se coloque en la escena respecto al sonido. El sonido se emite con un patrón elíptico, como se muestra en la figura 6.12.

Dentro del área roja, el sonido se oye con plena intensidad. Dentro del área amarilla, el sonido se desvanece según la distancia del origen. Los campos minFront, maxFront, minBack y maxBack especificados en el nodo Sound son las distancias que miden esto, así como la dirección.

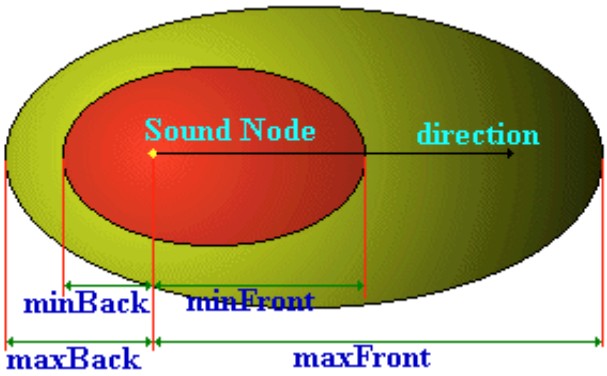


Figura 6.12. Zonas de distribución el sonido.

La sintaxis del nodo Sound es como sigue:

```
Sound {
  SFVec3f    direction      0 0 1
  SFFloat    intensity      1
  SFVec3f    location       0 0 0
  SFFloat    maxBack        10
  SFFloat    maxFront       10
  SFFloat    minBack        1
  SFFloat    minFront       1
  SFFloat    priority       0
  SFNode     source         NULL||
  SFBool     spatialize     TRUE
}
```

Los demás campos operan de la siguiente manera: direction es un vector que especifica la dirección a donde apunta el nodo sound. El campo intensity es el volumen del sonido. Un valor de 1 es volumen pleno, tal como se encuentra en el archivo original, y 0 es silencio. Situation es la posición de la fuente de sonido en el mundo. Los cuatro parámetros max/min de Front/Back especifican la forma de la elipse correspondiente al diagrama de arriba. Por ejemplo, para tener un sonido que sea igualmente audible en todas las direcciones, tendría el maxFront=maxBack, y minFront=minBack. Si no se quisiera que este sonido disminuyera con la distancia, se

tendría que poner todos los valores a la misma distancia, la cual deberá ser bastante grande como para cubrir un mundo entero. Si se quiere que el sonido vaya apagándose con la distancia, hay que poner los valores máximos a 10 y los valores min darán una proporción realista de caída.
El campo priority es la importancia del sonido. El navegador tendrá sólo un cierto número de canales de sonido disponibles para a él, y priority se utiliza para determinar cual de ellos se utilizará. Este campo toma valores de 0 a 1. Los sonidos de fondo de baja prioridad deben tener un valor de 0, y los prioritarios, sonidos cortos como timbres o lo que sea, deben tener un valor de 1.
El campo spatialize es usado por el navegador para determinar si debe especializar el sonido. Esto significa que lo hará sonar a través de los altavoces como si viniera de su origen, y entonces se oirá el sonido moviéndose cuando se mueva relativamente a él. Si este campo es FALSE, el navegador no hará esto. Esto es útil para la optimización y para la creación de sonidos ambientales.
El campo source (fuente, origen), es realmente importante. Puede contener un nodo AudioClip o un nodo MovieTexture. Si se usa MovieTexture, el navegador ejecutará el sonido del archivo de la película especificada. Esto puede ser útil para tocar archivos de sonido de MPEG, por ejemplo.

6.10.2 El nodo AudioClip

El nodo AudioClip especifica qué sonara por el nodo sound, y cuándo. La sintaxis es como sigue:

```
AudioClip {
  SFString   description     ""
  SFBool     loop            FALSE
  SFFloat    pitch           1.0
  SFTime     startTime       0
  SFTime     stopTime        0
  MFString   url             []
}
```

El AudioClip funciona así. El campo url es una lista de archivos en formato .WAV o General MIDI .MID tipo 1, que es útil para la representación compacta de música. El navegador tocará el primero de éstos que pueda cargar. Loop especifica si en el sonido se ejecuta en bucle, lo que es útil para los sonidos del ambiente y description es una descripción del sonido que el navegador puede desplegar si así se desea. El campo pitch es un multiplicador para la velocidad de la cinta, y en consecuencia del sonido. Un pitch de 1.0 es la velocidad normal. Un pitch de 0.5 sonará a la mitad de la velocidad, y transpone el sonido abajo una octava. Un valor de 2.0 tendrá el efecto opuesto, mientras duplica la velocidad del sonido, y eleva la velocidad una octava.
El startTime y stopTime son valores SFTime que especifican cuándo el sonido comenzará y se detendrá.

6.11 Geometría avanzada

6.11.1 IndexedFaceSet.

El nodo **IndexedFaceSet** es una colección de caras, que se pueden definir manualmente, y permiten construir objetos con cualquier forma. La idea básica es muy sencilla, y se muestra en la figura 6.13.

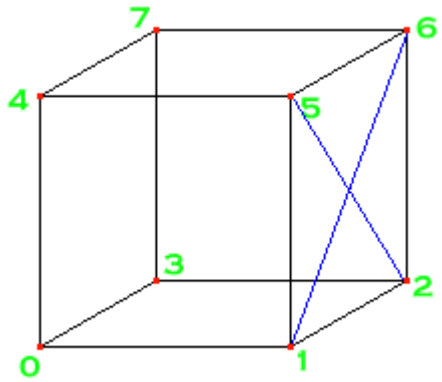


Figura 6.13 Colección de caras del nodo IndexedFaceSet.

Supongamos que queremos definir un cubo, o parte de un cubo. Primero necesitamos definir los ocho vértices, numerados del 0 al 7 en la figura 6.13. Ahora necesitamos definir las caras. Si se quiere definir la cara que tiene pintada la cruz azul, se debe utilizar los vértices 1,2,6 y 5. Y esto es todo lo que hay. Se hace lo mismo para cada cara y tendremos el cubo. Una palabra de aviso: si los vértices de una cara no son coplanares (esto es, no definen un área plana), entonces los navegadores provocaran errores y los resultados serán indefinidos. Es mejor usar caras triangulares, con solo tres vértices por cara, y de esta manera sé estará seguro de que siempre serán coplanares. Si se quiere más vértices por cara, sé debe ser cuidadoso.
La sintaxis actual de un nodo IndexedFaceSet básico es tal y como sigue:

```
IndexedFaceSet
  {SFNode     coord          NULL
   MFInt32    coordIndex     []
   SFBool     solid          TRUE
   }
```

El campo coord contiene un nodo Coordinate, que consiste en una simple lista de puntos 3D, tal y como se muestra en el ejemplo de abajo. Estos son los puntos en que consistirá el indexedFaceSet.
El siguiente campo es coordIndex. Es la lista de las caras. Para definir una cara, se debe introducir el índice de cada punto en la lista, seguido de un -1 para indicar que una cara ha acabado y que otra puede comenzar. El índice del punto es el número en que aparece en la lista. Si se está de frente a la cara, los valores de los índices deben introducirse en sentido contrario a las agujas del reloj. El único campo aparte de estos es solid (macizo). Si el objeto no está completamente cerrado, se deberá decir al navegador que se preocupe en mostrar también el "envés" de las caras, de otro modo no lo hará. El ejemplo inferior define una caja sin dos de sus caras, por lo que no es maciza.

```
  coord Coordinate
        {point [  -2 0 2, 2 0 2, 2 0 -2, -2 0 -2
                  -2 4 2, 2 4 2, 2 4 -2, -2 4 -2]
        }
  coordIndex
        [0 4 7 3 -1
         1 2 6 5 -1
         4 5 6 7 -1
         2 3 7 6 -1 ]
  solid FALSE
}
```

6.11.2 IndexedLineSet

Un nodo relativamente similar al IndexedFaceSet es el IndexedLineSet. Este nodo define un número de líneas que se dibujan entre una serie de coordenadas. Esto es útil para tener líneas muy delgadas, o para representar sólo la malla de un objeto. Se realiza exactamente de la misma manera que con los indexedFaceSet, excepto por el hecho de que no hay un campo solid y el campo coordIndex especifica el índice del punto entre cada línea, con un -1 entre cada una.

```
IndexedLineSet {
  SFNode     coord          NULL
  MFInt32    coordIndex     []
}
```

Las líneas son infinitamente delgadas, pueden colorearse, pero si el campo *color* del IndexedLineSet está vacío, el color se especifica por el campo *emissiveColor* del nodo Appearance.

6.11.3 PointSet

PointSet define un grupo de puntos en el espacio. Estos son dibujados con el tamaño que decide el navegador, y no hay manera de cambiar éste tamaño hasta el momento. Este nodo es idéntico al anterior, salvo por el hecho de que carece del campo coordIndex, dado que no hay conexión entre los puntos. Los puntos se dibujan de la misma manera que las líneas, y no son afectadas por las colisiones.

```
PointSet {
  SFNode     coord          NULL
}
```

6.11.4 ElevationGrid

Esto te permite crear un suelo irregular con distintas alturas. Esto se hace especificando un número de alturas, que se asignan a una cuadrícula. El tamaño de la cuadrícula se especifica con los campos xDimension y zDimension, xSpacing y zSpacing. Estos definen las dimensiones y espaciado entre los puntos en las direcciones X y Z. La lista de puntos comienza con aquel que está más alejado en la izquierda, procediendo hacia el que está más cercano a la derecha, cuando se mira hacia el eje -Z. si el campo solid es TRUE el objeto no podrá ser visto desde abajo. La sintaxis básica se muestra abajo.

```
ElevationGrid {
  MFFloat    height          []
  SFBool     solid           TRUE
  SFInt32    xDimension      0
  SFFloat    xSpacing        0.0
  SFInt32    zDimension      0
  SFInt32    zSpacing        0.0
}

ElevationGrid {
  xDimension 6
  zDimension 6
  xSpacing 5.0
  zSpacing 5.0
  height [ 1.5, 1, 0.5, 0.5, 1, 1.5,
         1, 0.5, 0.25, 0.25, 0.5, 1,
         0.5, 0.25, 0, 0, 0.25, 0.5,
         0.5, 0.25, 0, 0, 0.25, 0.5,
         1, 0.5, 0.25, 0.25, 0.5, 1,
         1.5, 1, 0.5, 0.5, 1, 1.5]
}
```

En este ejemplo, hay seis puntos las direcciones X y Z, y los puntos están alejados 5 metros en ambas direcciones. Esto genera una cuadrícula de 25 metros a cada lado. Los valores del campo son las alturas de cada punto. El primero se coloca en 0,0, y el último a +25, +25.

6.11.5 Extrusion

Esto toma una forma o sección en 2D, y la estira a lo largo de una ruta, llamada spine. Se puede alterar incluso el tamaño de cada punto, comprimiendo y expandiendo la sección. La sintaxis se muestra abajo:

```
Extrusion {
  SFBool     beginCap        TRUE
  MFVec2f    crossSection    [1 1, 1 -1, -1 -1, -1 1, 1 1]
  SFBool     endCap          TRUE
  MFVec2f    scale           1 1
  SFBool     solid           TRUE
  MFVec3f    spine           [0 0 0, 0 1 0]
  MFRotation orientation     0 0 1 0
}
```

El campo crossSection es una lista de puntos, en sentido contrario a las agujas del reloj, que definen la forma 2D de el objeto a extruir. Esta sección se alarga a través de la línea definida por el campo spine, que es una lista de puntos 3D. A cada punto de esta línea, se puede definir un factor de escala, en el campo scale, que toma por valor una lista de pares de números flotantes, uno por cada para las direcciones X y Z. El campo solid es igual que antes, y los campos beginCap y endCap definen si la extrusión posee o no una "tapa" al comienzo o al final. El campo orientation define una lista de rotaciones, una para cada punto de la línea. Esto permite un mayor control sobre la forma de la extrusión. Si no hay rotaciones suficientes para cada punto, los puntos restantes usan la última rotación de la lista. Así, este ejemplo, crearía una extrusión de un cuadrado a través de una línea de cuatro puntos con distintos factores de escala:

```
Extrusion
  {crossSection [1 1, 1 -1, -1 -1, -1 1, 1 1]
    spine [0 0 0, 0 2 0, 1 3 0, 2 3 0]
    scale [1 1, 1 0.5, 0.5 1, 0.5 0.5]
  }
```

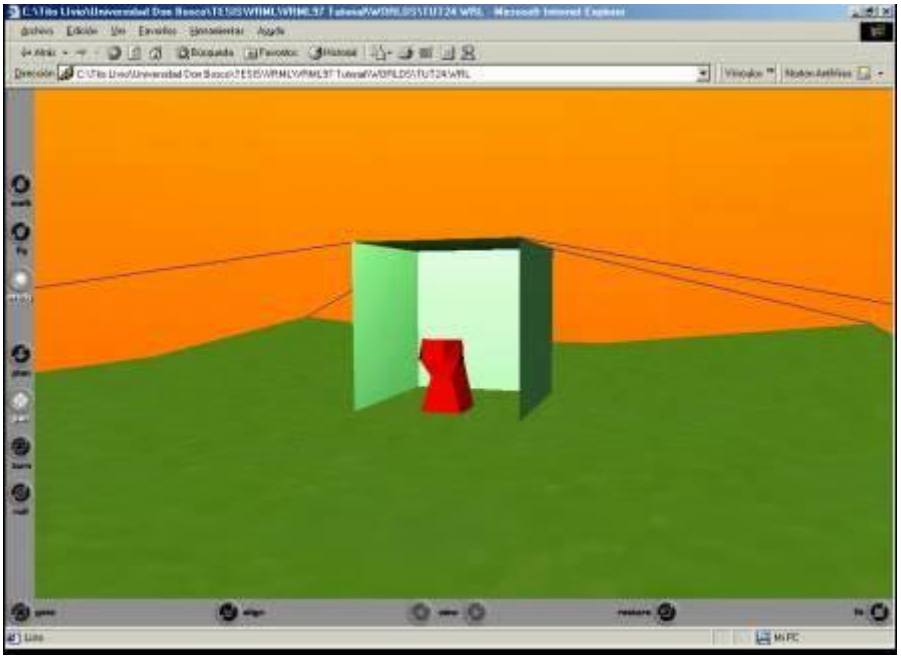


Figura 6.14 Mundo en base a nodos geométricos avanzados.

Se puede ver los efectos de todo este material observando el escenario de la figura 6.14. La "tienda" es un IndexedFaceSet, las "cuerdas" son IndexedLineSet, el suelo es un ElevationGrid y la "cosa" es un Extrusion.

6.12. Las normales, el color, y otras cosas para los objetos avanzados.

6.12.1 Normales

Un normal es un vector asociado a una cara, que por regla general permanece perpendicular a ella apuntando hacia fuera desde la cara visible. Las normales son usadas en visores VRML para cálculos de iluminación.

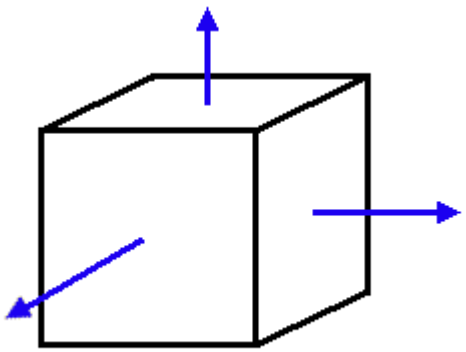


Figura 6.15 Normales.

En la figura 6.15, las normales de cada cara están dibujadas en azul. Apuntan hacia fuera, perpendicularmente a cada faceta.

El navegador usará las normales para cálculos de iluminación. Dependiendo de la dirección que tome la normal, el navegador sombreara la cara de manera diferente. Si dos caras tienen un ángulo entre ellas menor a una cantidad determinada, el navegador

suavizará la unión entre ellas, dando a la esquina la apariencia de una curva. Si el ángulo es mayor que esa cantidad, el navegador sombreará esta unión de manera afilada, dando una apariencia de facetas. Este ángulo es *creaseAngle*, que es un campo que aparece en las **Extrusion**, **ElevationGrid**, e **IndexedFaceSet**. Usándolo, se puede cambiar la apariencia de los objetos drásticamente, haciéndolos suaves o afilados, dependiendo de su valor.

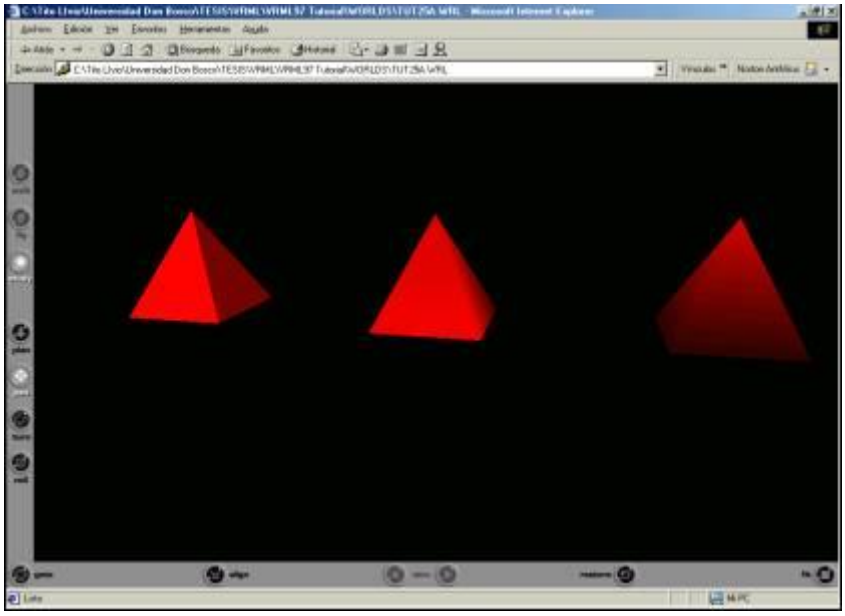


Figura 6.16 cambiando el campo creaseAngle para 3 objetos diferentes.

En el ejemplo de la figura 6.16, *creaseAngle* afecta la apariencia de los objetos. El de la izquierda tiene un *creaseAngle* de 0.5, (su valor por defecto), dándole una apariencia facetada. El del centro, tiene un *creaseAngle* de 2, dando una apariencia más suave en las junturas, y más afilada en la base. El de la derecha tiene un *creaseAngle* de tres, sombreando suavemente todas las junturas.

El navegador generará las normales por defecto para todos los tipos de nodo, incluidos los objetos avanzados. Sin embargo, estos objetos avanzados tienen un campo *normal*, que contiene un nodo Normal. Esto permite especificar las normales explícitamente, y así se pueden crear efectos de luz. Estas normales pueden ser especificadas tanto por vértice como por cara, usando el campo *normalPerVertex*. Si este campo está a TRUE, las normales se especifican por cada vértice. Si está a FALSE, son especificadas por cada cara. Si no especificas normales, el navegador lo hará él sólo asignándoles sus valores por defecto. Es la mejor solución la mayoría de las veces. Realmente sólo se usa "normal" cuando se trata de conseguir un efecto particular.
El nodo Normal contiene un conjunto de normales, que sólo se usa en el campo *normal* de un nodo.

```
Normal
    {exposedField   MFVec3f    vector          [ ]
    }
```

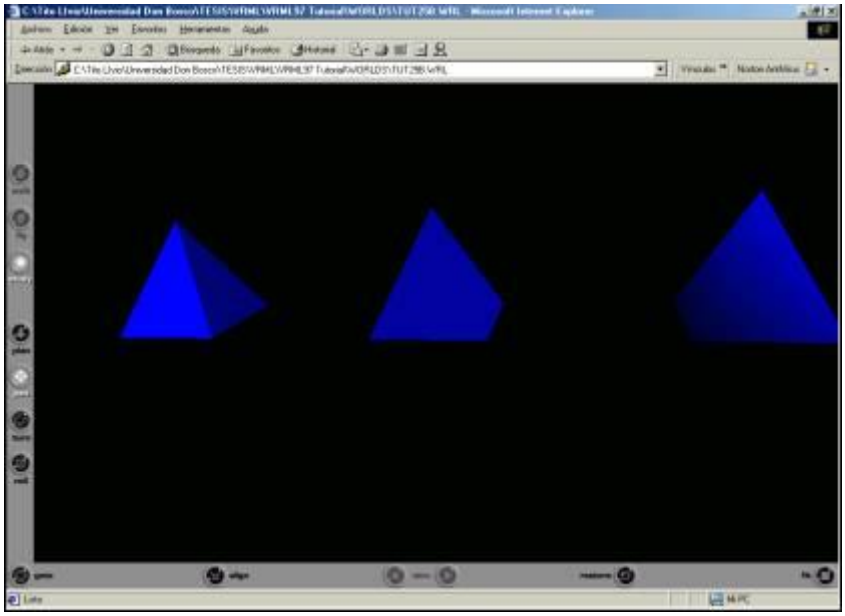


Figura 6.17 cambiando los valores del campo normalpervertex y del nodo Normal.

Este nodo sólo tiene un campo: *vector*. Contiene un conjunto de valores MFVec3f, que son los vectores de las normales para cada cara. La figura 6.17 muestra el efecto de aplicar normales por cara y por vértice.

El objeto de la izquierda permanece inafectado. El objeto del centro tiene sus normales aplicadas por cara, cambiando el modo en que es iluminada cada cara. El de la derecha del todo tiene sus normales aplicadas por vértice. En este caso, para dar una apariencia compleja y curvada a un objeto simple y afilado, por ejemplo, se puede hacer a un objeto cúbico tan suave como un cilindro con las normales adecuadas.

Los nodos de geometría contienen además el campo *ccw*, que especifica si los vértices de una cara se especifican en el sentido de las agujas del reloj. Esto afecta a la dirección de las normales establecidas por defecto, que se usan para cálculos de visibilidad. Si se observa directamente una cara con vértices en el orden contrario a las agujas del reloj, la normal apuntará hacia ti. En cambio, si se hace en el sentido de las agujas del reloj, apuntará hacia el otro lado, con lo que la cara no será visible. Disponer los valores de este campo incorrectamente, puede crear efectos muy interesantes, como objetos a los que les falta algún pedazo, el campo *ccw* está a TRUE por defecto.
El próximo campo que debemos considerar es *normalIndex*. Contiene una lista de números de caras o vértices, dependiendo de lo que hayamos puesto en el campo *normalPerVertex.* La primera cara o vértice especificada en el campo coordIndex/coord será la número 0, hasta el número total de caras o vértices menos 1. El campo *normalIndex* especifica qué normales se corresponden con qué vértices o caras. De esta manera, no se tiene por qué especificar las normales en el mismo orden en que estén dispuestos estos. En cambio, si dejas este campo en blanco, las normales serán aplicadas en este orden.

6.12.2 Colores

Exactamente de la misma manera que las normales, podemos aplicar diferentes colores por cara o por vértice. Esto puede hacerse con el campo *color*, que contiene un nodo **Color**, de la misma manera que el campo *normals*. Existe además un campo *colorIndex* y otro *colorPerVertex*, que hacen lo mismo que sus equivalentes en el campo *normals*.

Este ejemplo (figura 6.18) muestra como queda un objeto con colores especificados por vértice. El ejemplo es un cubo de colores RGB. La sintaxis del nodo **Color** se muestra abajo.

```
Color
  {exposedField   MFColor    color          []
  }
```

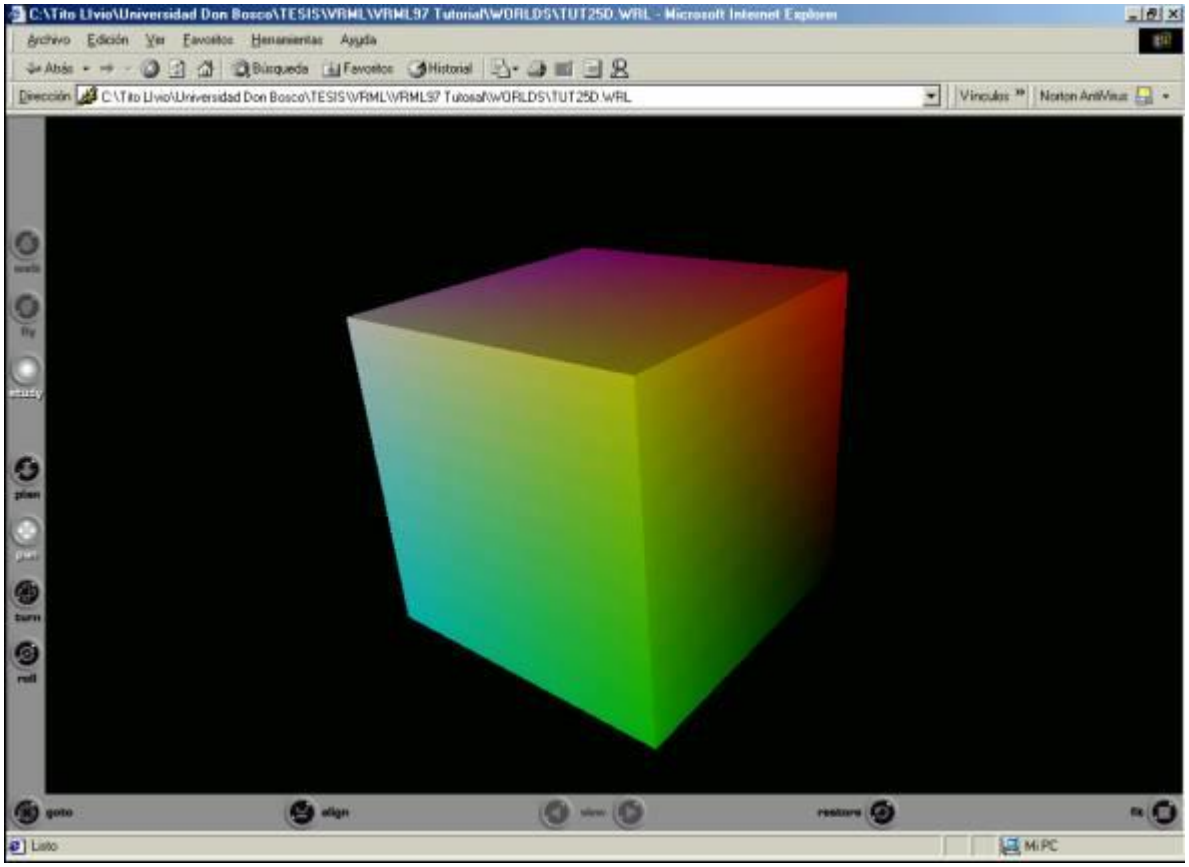


Figura 6.18 Colores especificados por vértices.

6.13 Iluminación en VRML

Lo primero que hay que decir es cómo funciona el modelo de iluminación del VRML. Todas las luces tienen los siguientes campos: *color*, *ambientIntensity*, e *intensity*. Cada luz tiene una intensidad (intensity) con un valor entre 0 y 1 que determina su brillo. También tiene una intensidad ambiental (ambient intensity), de nuevo entre 0 y 1, que determina cuanta luz contribuye al ambiente general de la escena. Por este motivo, cuantas más luces haya en la escena, más iluminado quedará el ambiente, lo que por otro lado es lógico. Una luz de ambiente (*Ambient*) es aquella que luce en todas las superficies de la escena, simulando provenir de otros objetos. Cada luz tiene además un color asociado, que sorprendentemente es el color que emite. La luz directa emitida por su fuente es calculada con intensity * color. La luz ambiental que contribuye a la escena es ambientIntensity * color. Cada fuente de luz tiene además un área de efecto, así que hay que recordar mantener las proporciones de las escenas. El método para hacer esto varía entre los diferentes tipos.

La mayoría de los navegadores VRML calculan la iluminación desde cada esquina hasta la cara, e interpolando las sombras entre esos vértices. Esto significa que si se tiene una cara gigantesca con una fuente de luz en el medio, y sus esquinas están muy distantes, NO se conseguira un foco de luz en el centro, si no que quedará uniformemente oscura. Esto es algo muy importante que hay que tener en mente. Otro punto importante es que las caras sólo se iluminarán si poseen un nodo Material. Los objetos con texturas no serán afectados por la iluminación. Otro detalle importante a recordar, es que por defecto los mundos VRML tienen el headlight del usuario en ON, con lo que iluminarán allá donde miren. Esto puede perjudicar seriamente el cuidado que se le ponga en la iluminación, así que hay que recordar quitarlo si se considera necesario. Esto se hace insertando la línea *headlight FALSE* dentro de un nodo NavigationInfo.

```
NavigationInfo
  {headlight FALSE
  }
```

6.13.1 Nodo DirectionalLight

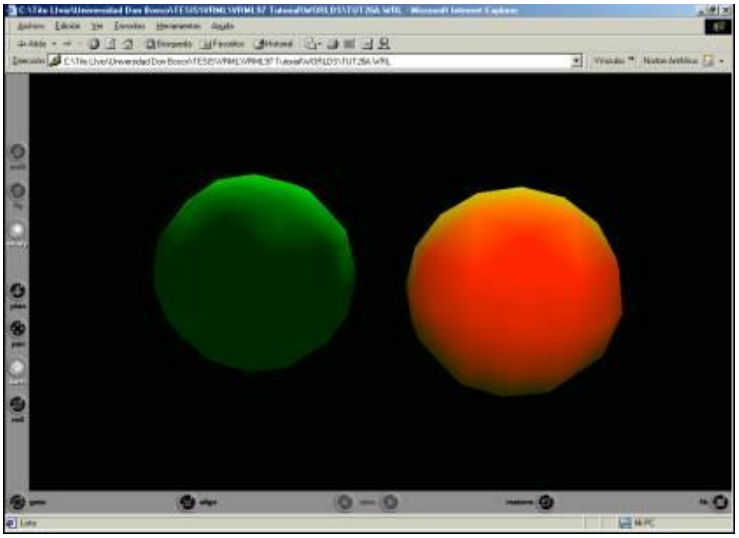


Figura 6.19 Esferas iluminadas con el nodo directionalLight.

Esta luz brilla desde una dirección uniformemente, como un haz de rayos paralelos. Da un efecto similar al del sol, donde todo es iluminado en la misma dirección. El nodo DirectionalLight no tiene una localización determinada en la escena, se limita a existir.
Ilumina todo aquello que sea children de su nodo padre, esto es, todo en su nivel jerárquico hacia abajo. La figura 6.19 muestra cómo funciona.
Ambas esferas son iluminadas por una luz verde desde atrás, pero una, a la derecha, está además iluminada por una luz roja desde el frente. Esto es así por que la luz verde está en el nivel más alto de la jerarquía, así que lo ilumina todo. La luz roja está agrupada en el mismo nodo que la esfera que hace brillar. La otra esfera está en un nivel distinto en el grafo de la escena, de manera que no queda iluminada. El siguiente diagrama lo ilustra todo.

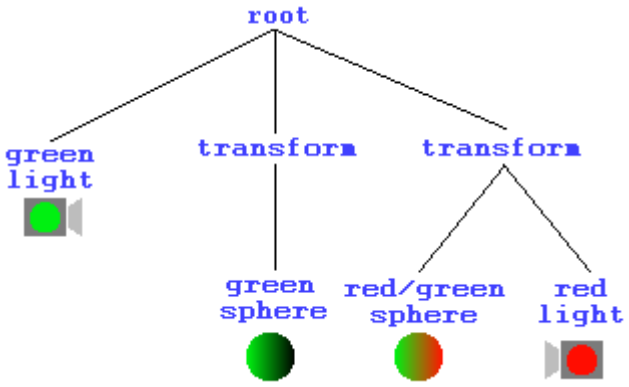


Figura 6.20 Jerarquía en la iluminación de las 2 esferas.

El nodo DirectionalLight tiene los campos básicos de *intensity*, *color*, y *ambientIntensity*. Tiene además un campo *on* cuyas valores varían de TRUE a FALSE, e indican si está o no encendido el foco. Tiene un campo *direction*, que es un vector que corresponde a la dirección hacia la que brilla. El valor por defecto es 0 0 -1, indicando que la luz apunta desde +Z hacia el origen descendiendo el eje -Z. La definición completa del **DirectionalLight** se muestra abajo.

```
DirectionalLight
 {exposedField  SFFloat    ambientIntensity  0
   exposedField  SFColor   color             1 1 1
   exposedField  SFVec3f   direction         0 0 -1
   exposedField  SFFloat    intensity        1
   exposedField  SFBool    on                TRUE
 }
```

6.13.2 Nodo PointLight.

Un PointLight es una luz que emana de un punto particular en el espacio, difundiéndose uniformemente en todas direcciones. Distintamente al DirectionalLight, este nodo tiene una posición en el espacio, especificada en su campo *location*. Este es el punto a partir del cual la luz emanará. Hay un campo que determina su área de efecto, que es *radius*. El radio es la distancia máxima dentro de la cual un objeto podrá ser iluminado. Dentro de este radio, la intensidad de la luz será afectada por una constante de atenuación. Estos son 3 valores de coma flotante (almacenados como un SFVec3f) que afectan a la iluminación según la siguiente fórmula:

$$\mathbf{1 / (attenuation[0] + attenuation[1]*r + attenuation[2]*r^2).}$$

Esto significa que los 3 valores en el campo attenuation se usan como coeficientes en una fórmula cuadrada por la intensidad de la luz a una distancia r. Está en decisión de uno experimentar con este coeficiente. El valor por defecto, **1 0 0**, no ofrece atenuación, dando un límite claro al efecto de la luz. Para hacer descender la iluminación linealmente, se debe tener una atenuación en la forma **0 x 0.**

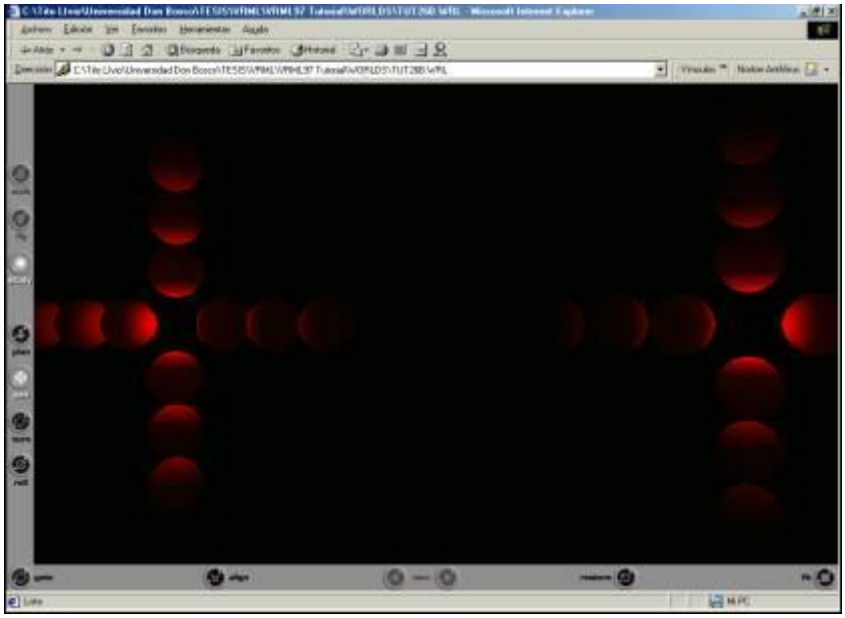


Figura 6.21 Iluminación con PointLight.

En la figura 6.21, el grupo de esferas de la izquierda está iluminado por unas constantes de atenuación de 0 1 0**,** dando una atenuación lineal. El de la derecha, tiene una atenuación no lineal de 0 0 1, provocando que la luz caiga en un r^2 del camino

```
PointLight {
  exposedField  SFFloat    ambientIntensity  0
  exposedField  SFVec3f    attenuation       1 0 0
  exposedField  SFColor    color             1 1 1
  exposedField  SFFloat    intensity         1
  exposedField  SFVec3f    location          0 0 0
  exposedField  SFBool     on                TRUE
  exposedField  SFFloat    radius            100
}
```

6.13.3 Nodo SpotLight

El último nodo de iluminación es SpotLight. Éste define una luz de foco. Tiene todos los campos habituales de los nodos de iluminación. Además tiene los campos *location*, *radius* y *attenuation*. Y hacen exactamente lo mismo que hemos descrito anteriormente, dan una posición, un radio de acción y una atenuación dentro de ese radio para la fuente de luz. Además, tiene otros campos. SpotLight tiene también su posición determinada en el espacio, al igual que PointLight**.** Eso sí, dado que es un tipo de luz que ilumina en una dirección particular, igual que tiene *location*, tiene también un campo *direction*, que define la dirección hacia la que apunta.

Además podemos especificar el tamaño del foco producido por otros dos campos, *beamWidth* y *cutOffAngle*. Estos son ángulos, que definen cómo de ancha será la proyección del foco, y cuán rápido deja de iluminar sus límites. *CutOffAngle* no tiene nada que ver con *attenuation. Attenuation* determina cómo cambia la luz en la dirección en la que brilla, y *cutOffAngle* determina como oscurece por los bordes del área iluminada. Por defecto, *beamWidth* es mayor que *cutOffAngle*, dando una apariencia afilada a los bordes del foco. Esto incrementa el trabajo de la representación considerablemente.

Este diagrama (figura 6.22) muestra como los campos *beamWidth* y *cutOffAngle* afectan la apariencia del área iluminada.

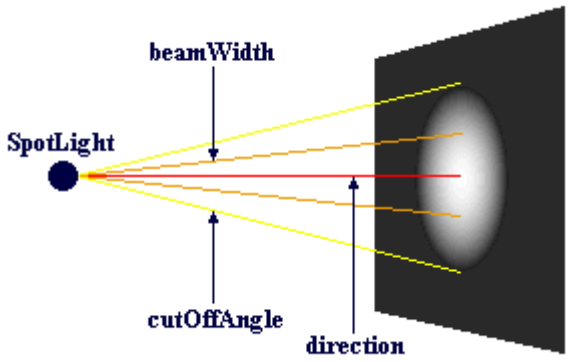


Figura 6.22 Los campos del nodo SpotLight.

```
SpotLight {
  exposedField  SFFloat   ambientIntensity  0
  exposedField  SFVec3f   attenuation       1 0 0
  exposedField  SFFloat   beamWidth         1.570796
  exposedField  SFColor   color             1 1 1
  exposedField  SFFloat   cutOffAngle       0.785398
  exposedField  SFVec3f   direction         0 0 -1
  exposedField  SFFloat   intensity         1
  exposedField  SFVec3f   location          0 0 0
  exposedField  SFBool    on                TRUE
  exposedField  SFFloat   radius            100
}
```

6.14 NavigationInfo

Permite personalizar varios parámetros del navegador relativos al usuario. Muchos navegadores permiten que el propio usuario los cambie, pero puede decirle como debe dejarlos al entrar en tu escena. La definición es tal como sigue:

```
NavigationInfo {
  eventIn       SFBool    set_bind
  exposedField  MFFloat   avatarSize      [0.25, 1.6, 0.75]
  exposedField  SFBool    headlight       TRUE
  exposedField  SFFloat   speed           1.0
  exposedField  MFString  type            "WALK"
  exposedField  SFFloat   visibilityLimit  0.0
  eventOut      SFBool    isBound
}
```

El nodo tiene dos eventos, *set_bind* e *isBound*. Esto es así por que se trata de un nodo bindable**.** Es un nodo de raíz, de modo que puede ir en la cabeza de la jerarquía del grafo de la escena, de un nodo children.

Las cosas que se pueden manipular están representadas por los campos de nodo. El primero que veremos es *headlight*. El usuario puede cambiarlo, pero se puede especificar el que esté por defecto. Si no se especifica lo contrario, un archivo VRML tiene por defecto su valor a ON, lo cual es útil para escenas sin iluminar. Pero, en mundos elaboradamente iluminados, resulta innecesario e irritante. Así que se puede apagar poniéndola a FALSE en el nodo NavigationInfo. Se puede especificar además un límite de visibilidad (*visibilityLimit*) a tu mundo. Esta es la distancia desde el usuario tras la cual nada será dibujado en pantalla. Se puede usar para aumentar la velocidad de ejecución en escenarios muy grandes. Si está determinado a 0 0 0 (tal y como viene por defecto), no habrá limite de visibilidad. El siguiente campo es *speed*. Es un

multiplicador que el navegador usa para determinar la velocidad del usuario mientras se mueve. Como algunos navegadores permiten velocidades variables, esto es una indicación al navegador para indicar cuán rápido puede moverse el usuario. Un valor de 3.0 permitirá al usuario moverse tres veces más rápido de lo normal. Una velocidad de 0.5 dividirá su velocidad.
El campo *type* es el que determina como se mueve el usuario a través de tu mundo. El campo puede tomar diferentes valores. Estos son:

- **WALK**
  Donde el usuario camina normalmente siendo afectado por la gravedad,
- **FLY**
  Donde el usuario se mueve sin ser afectado por la gravedad,
- **EXAMINE**
  Donde el usuario permanece inmóvil, pero puede rotar el mundo para verlo desde distintos ángulos.
- **NONE**
  Donde el usuario no tiene control de movimiento en absoluto.

Se pueden ver distintos estilos en los cuatro escenarios. Podemos darnos cuenta de que WALK y FLY son exactamente lo mismo, con la diferencia de que WALK nunca te levantará del suelo, mientras que FLY sí.
La elección del tipo de navegación da una gran flexibilidad en cómo puede usarse el VRML. Con el campo a WALK o FLY, se podrá tener la clásica sensación de mundo virtual. Con EXAMINE se podrá tener un modelo de un coche u otro producto, de manera que pueda examinarse fácilmente. Finalmente, con NONE, se podrá poner al usuario a mirar lo que ponen de frente. Puede ser usada para películas 3D, animaciones, o anuncios.
El campo *avatarSize*. Permite determinar el espacio físico de la presencia del usuario (o *avatar*) en el mundo. Tiene tres valores de escala. Estos representan, en orden, al avatar:

- Radio,
- Altura de los ojos,
- Altura de las rodillas.

El avatar se trata como una forma cilíndrica para la detección de las colisiones. El cilindro tiene unas proporciones tal y como se muestra en el diagrama inferior:

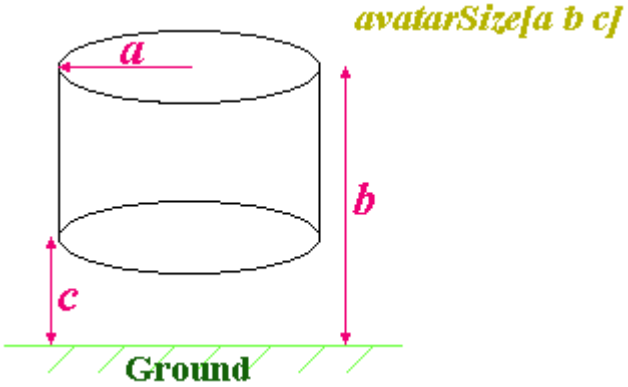


Figura 6.23 Espacio avatar.

El radio y la altura total son bastante evidentes, pero la altura de las rodillas requiere una pequeña explicación. Cuando el usuario está en modo WALK, puede subir y bajar escaleras, pero sus pasos estarán condicionados por la altura de sus rodillas. El avatar

sólo podrá subir escalones de altura inferior a la de sus rodillas. Esto te permite crear escalones adecuados, mientras siguen existiendo muros que el usuario no puede escalar.

6.15 Geometría orientada al observador

6.15.1 El nodo Billboards

El nodo **Billboard**. Realmente es un nodo **Transform** que rota automáticamente para estar de cara al usuario. No es necesario decir lo útil que resulta para carteles de texto, sprites, etc.

```
Billboard {
  eventIn       MFNode     add Children
  eventIn       MFNode     removeChildren
  exposedField  SFVec3f    axisOfRotation    0 1 0
  exposedField  MFNode     children          []
  field         SFVec3f    bboxCenter        0 0 0
  field         SFVec3f    bboxSize          -1 -1 -1
}
```

Como todos los nodos de agrupación, posee los eventos *addChildren*, *removeChildren*, *bboxCenter* y *bboxSize*. Tiene además un campo *children*, que contiene los objetos que quieres en tu billboard. De modo que si se quiere un cubo que muestre siempre la misma cara al usuario, se coloca en este campo.El campo *axisOfRotation*. Este es el eje sobre el que va a rotar el nodo. El valor por defecto es 0 1 0, que es el eje Y. Esto es lo que normalmente se querría para que el usuario pueda ver el billboard. Pero si se tiene un objeto con un ángulo dado, se puede querer un ángulo diferente. El caso especial para el billboard es el 0 0 0. Normalmente, este no es un eje válido, pero en este caso le dice al nodo que puede rotar sobre CUALQUIER eje. Normalmente, cuando el nodo gira alrededor de un ángulo, si se aparta de él dejará de verlo claramente. Si tienes un eje de 0 0 0, apuntará al usuario, no importa donde esté.

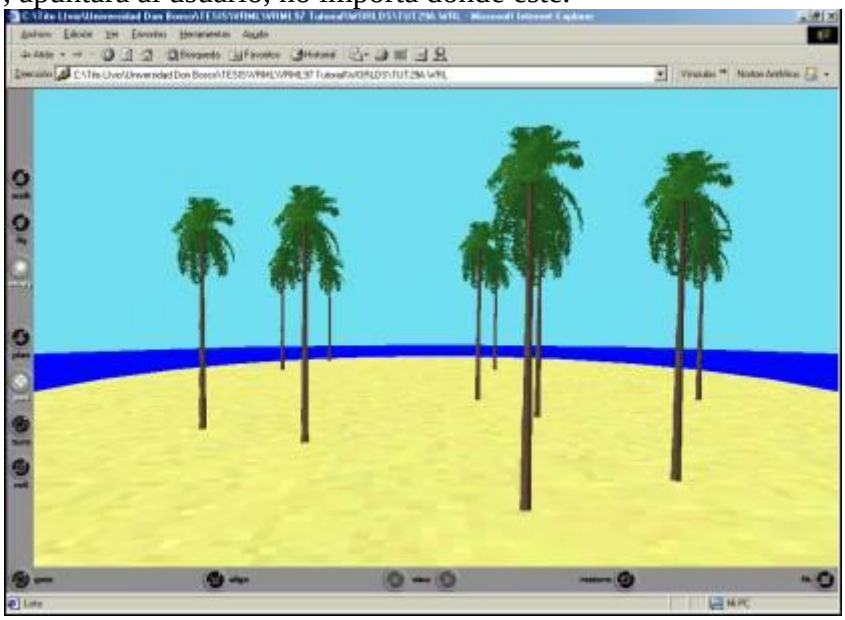


Figura 6.24 Nodos Billboard.

En la figura 6.24 se da una demostración. Todos los árboles son **Billboard** que giran respecto a su eje Y, de modo que siempre mirarán al usuario.

6.15.2 Más sobre el nodo Transform

El nodo Transform al completo es más complejo que todo eso. La definición completa se muestra aquí:

```
Transform {
  eventIn       MFNode     addChildren
  eventIn       MFNode     removeChildren
  exposedField  SFVec3f    center              0 0 0
  exposedField  MFNode     children            [ ]
  exposedField  SFRotation  rotation             0 0 1 0
  exposedField  SFVec3f    scale               1 1 1
  exposedField  SFRotation  scaleOrientation    0 0 1 0
  exposedField  SFVec3f    translation         0 0 0
  field         SFVec3f    bboxCenter          0 0 0
  field         SFVec3f    bboxSize           -1 -1 -1
}
```

Como se puede ver este nodo de agrupamiento tiene además los eventos *addChildren*, *removeChildren*, *bboxCenter* y *bboxSize*. Ya hemos visto *rotation*, *scale*, *translation* y *children.* Los campos *center* y *scaleOrientation* transforman el sistema de coordenadas local del nodo mientras las otras transformaciones se aplican.
El campo *center* afecta a las transformaciones de rotación y escala. Define el origen local de la transformación. Así que si se define un centro de 5 0 0, las rotaciones y escalas serán relativas a este punto. El objeto rotará alrededor de ese punto, lo que es equivalente a trasladar el objeto por el inverso de ese centro, rotarlo, y volver a trasladarlo de vuelta. Esto afecta también a escalas, de modo que si se escala algo, pero se mueve el origen de la escala, afectará todas las coordenadas del objeto relativas al *center* en vez de al origen local del objeto.
El campo *scaleOrientation* tiene también efecto en la escala, sólo que esta vez es una rotación lo que se aplica en vez de una traslación. Esto permitirá escalar en las direcciones que mejor te parezca. Con *scale*, sólo se puede escalar sobre los ejes, lo cual es algo limitado. Usando *scaleOrientation se* pueden rotar las direcciones de escala para un objeto. Tiene el mismo efecto que aplicar la rotación inversa, escalar, y rotar el objeto de vuelta. Esencialmente, rota los ejes de la escala con el campo *scaleOrientation*.

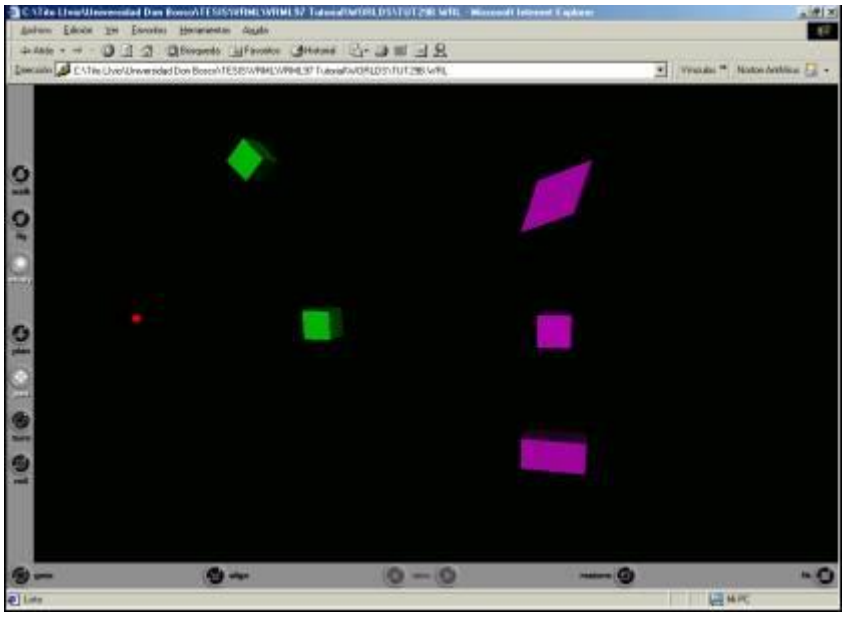


Figura 6.25 Más sobre el nodo Transform.

En la figura 6.25, a la izquierda, tenemos una rotación sobre un punto arbitrario. El punto rojo marca el centro, sobre el cual ha sido rotada la caja verde. Las cajas púrpuras han sido escaladas. El centro es una caja normal, sin escalar. La caja de abajo ha sido escalada con un valor de 2 en la dirección X. La de arriba también, pero con un *scaleOrientation* de 0 0 1 0.78. Esto la hace rotar sobre el eje X 45 grados en el sentido contrario a las agujas del reloj, dándole una extraña apariencia.

6.16 Elementos esenciales de animación en VRML

Casi todos los nodos poseen **eventIns** y **eventOuts** (eventos de entrada y de salida), y muchos tienen **exposedFields.** Es así como los nodos se comunican con el exterior. Los eventIns son semejantes a receptores que escuchan los mensajes, llamados eventos, del exterior y los toman para procesarlos. Los eventOuts son transmisores que envían los eventos producidos por el nodo al exterior. Los exposedFields son una combinación de ambos. Se comportan como un campo ordinario, y poseen un eventIn llamado *set_nombredecampo* y un eventOut llamado *nombredecampo_changed.* En general, las partes *set_* y *_changed* no son necesarias, se puede usar simplemente el *nombredecampo* del exposedField y en ejecución el navegador sabrá lo que se le quiere decir.
Ahora, todo esto es inútil si los eventOuts van a parar al espacio vacío. No van a ningún lado de esta manera. Se necesitara conectar los nodos a través de **ROUTE (rutas)**. Éstos son como cañerías que encauzan los eventos de un eventOut hacia un eventIn.

Por ejemplo, si se tiene dos nodos, un TouchSensor y un nodo Sound, declarados DEF SENSOR y SONIDO respectivamente, se puede dirigir el eventOut del *touchTime* del nodo TouchSensor al eventOut *startTime* del nodo Sound**,** (por ejemplo para hacer sonar un click al pulsar con el ratón). Usaríamos la siguiente línea de código:

ROUTE SENSOR.touchTime TO SONIDO.startTime

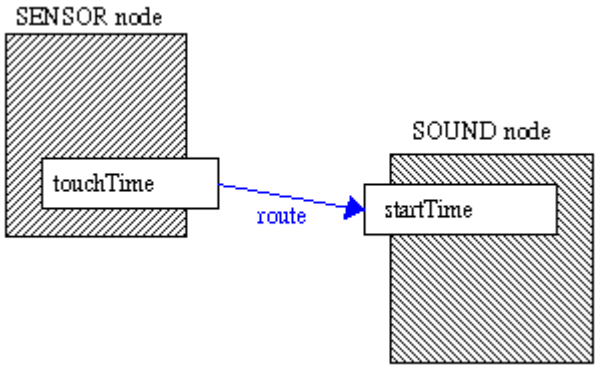


Figura 6.26. Conexión de 2 nodos a través de ROUTE.

Cuando se pulsa con el ratón sobre el TouchSensor, el sonido se ejecutará. Como práctica común, todas las declaraciones de ROUTE se agrupan al final del archivo en un grupo aparte. No pueden ir dentro de cualquier nodo, y estos están completamente aparte de las cosas que enrutan. También se debe saber que las Route pueden conectar sólo eventos del mismo tipo, por lo que no se puede conectar un SFTime, por ejemplo, a un SFBool. Si se necesita hacer esto, se debe hacer un pequeño script que convierta un dato de un tipo a otro determinado.

6.16.1 Generando Eventos

Los eventos son mensajes que ligan los elementos de la escena. Todo lo que se mueve o interactúa en VRML lo hace debido a los eventos. Son la clave de todo este asunto. Un evento posee dos partes: el propio mensaje, que es un valor o dato de un cierto tipo, y un time stamp, o estampa de tiempo. El valor del mensaje puede ser de cualquier tipo, por ejemplo SFTime, SFString, MFNode, cualquier cosa.

La estampa de tiempo es un valor especial sobre el que no se tiene ningún control. Es un valor que corresponde al momento preciso en que el evento se produjo, no cuando el evento ocurrió para ser generado por el nodo. Los valores reales no son importantes, sólo cómo se relacionan con otros valores. Un evento con una estampa de tiempo posterior se define como si ocurriera después de otro con una estampa de tiempo anterior. En general, se procesan los eventos en orden del incremento de la estampa de tiempo. La estampa de tiempo no está disponible al programador, y sólo se usa internamente por el navegador.

Si un evento se genera, y esto causa otro evento en otro nodo, y así continuamente una y otra vez, se dice que se está produciendo una *cascada de eventos*. Todos los eventos en cascada tendrán el *mismo* time stamp. Es decir, si el navegador emplea o consulta él time stamp, creerá que todos los eventos se han producido a la vez. Con un fan-in en un eventIn, si dos eventos se reciben con la misma estampa, los resultados son impredecibles. Pueden ser ignorados ambos, o realizarse ambos, o sólo uno de ellos... no hay ninguna manera de saberlo, y variará según el navegador que se emplee. Sin embargo, y afortunadamente, casi la única manera de hacer esto es tener un fan-in dentro de una sola cascada de eventos, para que, con un estudio cuidadoso de la escena, estas situaciones puedan evitarse.

Los bucles también pueden ocurrir en cascadas de eventos dónde un evento causa otro, que a su vez, al volver, causa el primero. Esto también debe evitarse en la mayoría de los casos. Sin embargo, si es necesario, puede usarse. El modelo de ejecución permitirá sólo 1 evento con un timestamp particular enviado desde cada eventOut, por lo que sí hay un bucle actuando no se ejecutará continuamente, sino sólo una vez. Esto es porque un bucle debe ser parte de una cascada de eventos, y dado que el segundo evento generado por el nodo tendrá la misma estampa de tiempo que el primero, no se enviará. Así que, se permite el uso de bucles, pero probablemente no funcionarán como se esperan.

Los eventos iniciales (eventos no causados por otro en una cascada) sólo puede generarse por nodos de tipo sensor y nodos Script. Otros nodos pueden generar eventos sólo si reciben uno. Esto significa que los sensores y Scripts son la llave de la animación y de la interacción en VRML.

Los nodos que finalmente llevan a cabo la animación real son los nodos interpoladores y Switch, así como el *exposedField* en otros nodos. Hasta llegar a ellos hay sensores, Scripts, y cualquier otro elemento que forme parte de una estructura compleja de animación. El modelo VRML de animación es, de hecho, muy poderoso, y sus posibilidades son prácticamente ilimitadas.

6.16.2 Estructura de una animación

Así, el modelo de la figura 6.27 parece bastante complejo. Un **TouchSensor** envía un solo evento *touchTime* (SFTime) a un nodo **Script** siempre que se active. El nodo **Script** convierte el SFTime en un SFBool y manda el resultado (true o false) al campo *enabled* del **TimeSensor** que comenzará entonces (o detendrá) su marcha. Mientras el

TimeSensor está habilitado (esto es, su campo enabled devuelve true) genera eventos continuamente para hacer variar el ciclo **ColorInterpolator** del *diffuseColor* de un objeto. Estos eventos del **TimeSensor** también llevan un **PositionInterpolator** que cambia el campo *translation* del nodo **Transform**.

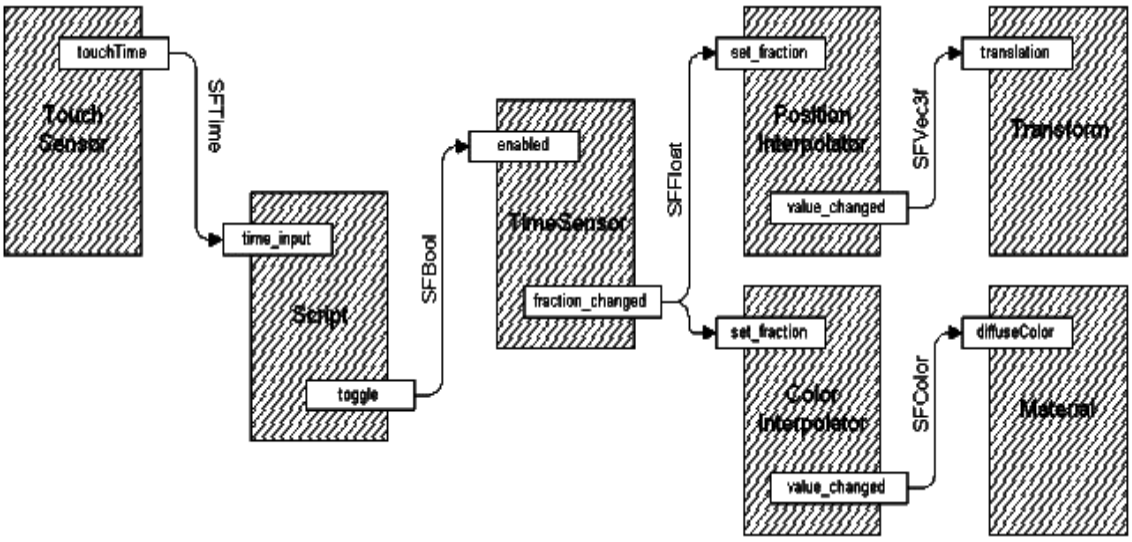


Figura 6.27. Estructura de una animación.

Hay nodos que no se han visto en el ejemplo anterior pero se tocaran más tarde, pero ahora solo basta concentrarse en las conexiones y el flujo de datos. Éstos son aquí los elementos esenciales. Ahora se sabe conectar los nodos entre sí para hacer comportamientos complejos, y cómo y por qué estos comportamientos se producen.

Ahora que se conoce la arquitectura básica del sistema de animación VRML, vamos a empezar cubriendo los nodos que se usaran normalmente para la interacción y animación de tus mundos.

Hay tres clases principales de nodos sobre los que vamos a aprender: Los *sensores*, *Interpoladores* y *Scripts*. Éstos, a su vez, pueden ser divididos en subcategorías.

6.16.3 Usando los sensores medioambientales

Los sensores medioambientales no aceptan la entrada directamente del usuario, pero en cambio captan eventos medioambientales, como el paso de tiempo, la posición del usuario, y otras cosas útiles.

6.16.3.1 TimeSensor

Es básicamente un timer (cronómetro). Es único en VRML, no tiene ninguna posición en el mundo, y ninguna geometría asociada. Simplemente es un cronómetro abstracto, sentado tranquilamente mientras cuenta. También es uno de los nodos más importantes en la animación de VRML. Puede usarse para generar eventos regulares, proporcionar eventos cronometrados, o manejar los nodos interpoladores. La sintaxis es como sigue.

```
TimeSensor {
  exposedField  SFTime    cycleInterval   1
  exposedField  SFBool    enabled         TRUE
  exposedField  SFBool    loop            FALSE
  exposedField  SFTime    startTime       0
  exposedField  SFTime    stopTime        0
  eventOut      SFTime    cycleTime
```

```
  eventOut      SFFloat    fraction_changed
  eventOut      SFBool     isActive
  eventOut      SFTime     time
}
```

En primer lugar esta el campo *cycleInterval*. Es bastante auto explicativo: devuelve el tiempo que el timer correrá antes de restablecerse. *Enabled* (habilitado) es también un campo bastante obvio, y muy útil para activar y detener los timers. El campo *loop* especifica si el timer se ejecutará continuamente o si simplemente lo hará una vez. Si está a true, se generará un evento cada *cycleInterval*. De otra manera sólo se generará uno, después del *cycleInterval*. Los campos *startTime* y *stopTime poseen* valores SFTime que especifican cuándo comienza a contar el cronómetro y cuándo se detiene, igual que para el nodo Sound.
Ahora, los eventos. Son partes muy importantes del TimeSensor. El primero es el *cycleTime*. Este evento se envía cada vez el cronómetro alcanza el *cycleInterval*, tanto si se repite con el loop como si no. El valor enviado es el tiempo actual. Así, si se tuviera un TimeSensor repitiéndose continuamente con un *cycleInterval* de 1 segundo, el evento del *cycleTime* se enviaría todos los segundos con un valor del tiempo actual (qué aumentaría en 1 cada vez).

Esto es útil para eventos regulares y señales intermitentes. Para manejar animaciones continuadas, como las producidas con nodos Interpoladores, necesitamos un riego continuo de señales. Esto se logra mediante el eventOut del *fraction_changed*. Esto genera eventos tan rápido como puede (aunque no hay ninguna garantía sobre la regularidad de esos eventos) y devuelve un valor SFFloat que es el fragmento del *cycleInterval* que está actualmente completo. Por ejemplo, si tuvieras un *cycleInterval* de 10 segundos, esto es lo que un *fraction_changed* devuelve en un bucle TimeSensor.

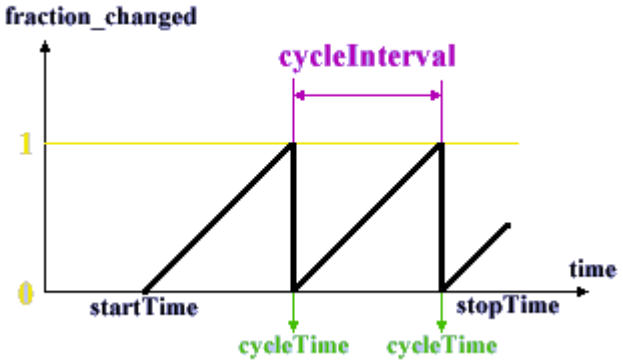


Figura 6.28

6.16.3.2 VisibilitySensor.

Es más un sensor para la interacción; en él se define una caja invisible que envía eventos cuando entra y sale del campo de visión del usuario. Si la caja entra el campo de visión, el evento *isActive* envía un valor de TRUE, y el evento *enterTime* el momento de entrada. Si sale, el evento *isActive* envía un valor de FALSE. Los campos *exitTimecenter* y *size* definen el tamaño de la caja. La definición completa se muestra a continuación:

```
VisibilitySensor {
  exposedField  SFVec3f   center        0 0 0
  exposedField  SFBool    enabled       TRUE
  exposedField  SFVec3f   size          0 0 0
  eventOut      SFTime    enterTime
  eventOut      SFTime    exitTime
  eventOut      SFBool    isActive
}
```

VisibilitySensor será muy útil para optimizar las escenas. Tener un número grande de animaciones que corren al mismo tiempo puede ser un trabajo muy duro para la CPU, y reducirá horriblemente la velocidad de los mundos. Usando VisibilitySensor para detener animaciones que están fuera del campo de visión de los usuarios, se podrá acelerar las animaciones de los mundos tranquilamente hasta el límite que se considere conveniente. Esto puede hacerse sencillamente enviando el evento *isActive* al campo *enabled* del TimeSensor apropiado.

6.16.3.3 ProximitySensor

Un ProximitySensor es muy similar a un VisibilitySensor, sólo que genera eventos *isActive* cuando el usuario entra o sale de la caja definida por el nodo. Los eventos *enterTime* y *exitTime* funcionan como antes, como el evento *isActive.* Mientras el usuario esté dentro del ProximitySensor se lanzaran eventos, siempre que su posición u orientación relativas al ProximitySensor cambien, a través de los eventOut position_changed y orientation_changed. Éstos contendrán el valor de la nueva posición u orientación del avatar, cualquiera que sea.

Recordar, éstos están en el sistema de la coordenada local del ProximitySensor, que será relativo al centro del sensor.

```
ProximitySensor {
  exposedField  SFVec3f     center              0 0 0
  exposedField  SFVec3f     size                0 0 0
  exposedField  SFBool      enabled             TRUE
  eventOut      SFBool      isActive
  eventOut      SFVec3f     position_changed
  eventOut      SFRotation  orientation_changed
  eventOut      SFTime      enterTime
  eventOut      SFTime      exitTime
}
```

6.16.4 Interpoladores para la animación.

Son muy importantes para las animaciones, y en cambio son muy sencillos. El concepto general es que se usan para cambiar determinados valores según pasa el tiempo. Dependiendo del valor que se desee cambiar, se usará uno de los seis tipos distintos de interpoladores que existen. Dado que son muy similares, los cubriremos de pasada.
Se usará interpoladores cuando se quiera cambiar el valor de un campo cada determinado tiempo. Un interpolador toma la señal de un **TimeSensor**, y realiza una interpolación lineal entre un juego de valores llamado *keyValues.* Para cada keyValue hay una *key* (clave) que es una fracción de 0 a 1. Ahora, como se recordará, un **TimeSensor** envía eventos *fraction_changed* tan regularmente como puede. Esto puede dirigirse al eventIn *set_fraction* de un interpolador para llevarlo al punto correcto del ciclo de la interpolación. Así, se dirige un **TimeSensor** con un *cycleInterval* de 10 a un interpolador, este pasará por cada clave cada 10 segundos.

Cada vez que un interpolador recibe un eventIn, genera un eventOut con el valor interpolado apropiado. Esto puede dirigirse a algún campo del tipo apropiado para cambiar su valor. Así, la cadena de la animación global para un interpolador se parece el diagrama de la figura 6.29. Este diagrama usa un **OrientationInterpolator** y un nodo **Transform** para demostrar el esquema del ROUTE.

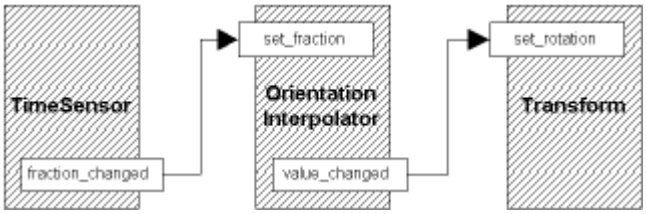


Figura 6.29 Utilizando OrientationIterpolador.

6.16.4.1 Tipos de Interpoladores.

Pueden ser usados en cualquier sitio, al principio del archivo, dentro de un nodo **Transform**, etc. Como están completamente involucrados en el mecanismo de la animación, pueden entrar en cualquier parte de la jerarquía de la escena, no importa dónde.
Todos los nodos tienen los mismos campos, pero con tipos diferentes para el *keyValue* y el eventOut *value_changed.* Así que primero tendremos un eventIn *set_fraction* que recibe los eventos de un **TimeSensors** o de otra parte. Este lleva a la fase actual del ciclo de la animación, y determina hasta qué punto de la interpolación hemos llegado.

El próximo campo es *key*. Es una sucesión de números de 0 a 1 correspondiendo a los puntos clave en la animación. Así, si en fracción 0 se quiere un cierto valor de posición, y se quiere ir cambiando para las fracciones 0.5 y 1.0, se debe tener un campo *key* de [0, 0.5, 1], en el que cada número representa un fragmento de cada keyframe.

Cada key (o clave) tiene una entrada correspondiente en el campo *keyValue*, por lo que, si seguimos el ejemplo anterior (usando un **ScalarInterpolator** que interpola valores SFFloat), podríamos tener un campo *keyValue* de [0.0, 10.0, 0.0].

En fragmento 0, el valor enviado sería 0.0, en 0.5 sería 10.0, y en 1 estaría de nuevo a 0.0. Entre estos *key*s el valor se interpola linealmente, por lo que en la fracción 0.25, el valor estará en 5.0. Este valor es la salida de datos a lo largo del eventOut siempre que se reciba un evento *set_fraction*.

Los interpoladores permiten especificar los "fotogramas" clave en una animación y el navegador gestiona todos los pasos intermedios para generar la animación. Esto permite una definición concisa y simple de la animación. Una cosa importante que debes recordar es que si quieres una animación cíclica, deberás tener el mismo valor en la fracción 1 que en la 0. Si no, la animación saltará bruscamente de la última clave a la primera.

```
ColorInterpolator
 {eventIn       SFFloat  set_fraction
  exposedField  MFFloat  key              [ ]
  exposedField  MFFloat  keyValue         [ ]
  eventOut      SFFloat  value_changed
}
```

El primero es **ColorInterpolator.** Como es evidente, interpola valores triples de color RGB. Los colores se declaran en el campo *keyValue* de la misma manera en que especificarías cualquier campo MFColor. Si quisiéramos un color que pasara de verde a

rojo, pasando por azul, deberíamos usar algo como lo siguiente (piensa en que esta animación, de ser cíclica, tendría un salto brusco del final al principio):
key [0, 0.5, 1]
keyValue [0 0 1, 0 1 0, 1 0 0]

El resto de los nodos interpoladores son el mismo nodo continuamente, pero con diferentes valores para el campo *keyValue.* De todos modos, vamos a detenernos en ellos.

```
CoordinateInterpolator
 {eventIn          SFFloat  set_fraction
  exposedField   MFFloat  key                [ ]
  exposedField   MFVec3f  keyValue          [ ]
  eventOut         MFVec3f  value_changed
}
```

Este, en lugar de enviar un solo valor, envía muchos al mismo tiempo. Esto es para que se pueda interpolar muchas coordenadas (un **IndexedFaceSet** entero) de una sola vez. Para hacer esto, el número de coordenadas del campo *keyValue* tiene que ser un múltiplo exacto del número de *key*s. Este número de coordenadas será lanzado al mismo tiempo. Pensemos que debemos guardar el orden de las coordenadas cada vez, o el objeto resultará francamente extraño.

```
NormalInterpolator {
eventIn       SFFloat  set_fraction
exposedField   MFFloat  key              [ ]
exposedField   MFVec3f  keyValue          [ ]
eventOut      MFVec3f  value_changed
}
```

Como antes, este nodo envía muchos normales al tiempo, por distintos motivos. La salida de datos del nodo deben ser varios valores de normales, para poder manejar todos los normales de un solo objeto a la vez.

```
OrientationInterpolator {
  eventIn       SFFloat    set_fraction
  exposedField   MFFloat    key           [ ]
  exposedField   MFRotation  keyValue       [ ]
  eventOut      SFRotation  value_changed
}
```

Esto interpola valores de rotación, y es conveniente dirigirlo al *set_rotation* de un nodo **Transform**.

```
PositionInterpolator {
  eventIn       SFFloat  set_fraction
  exposedField   MFFloat  key              [ ]
  exposedField   MFVec3f  keyValue          [ ]
  eventOut      SFVec3f  value_changed
}
```

De nuevo, esto interpola valores de posiciones tridimensionales, y puede dirigirse al *set_translation* de un **Transform.** Sin embargo, puede ser dirigido hacia cualquier eventIn que acepte valores SFVec3f.

```
ScalarInterpolator {
  eventIn       SFFloat  set_fraction
  exposedField  MFFloat  key             []
  exposedField  MFFloat  keyValue        []
  eventOut      SFFloat  value_changed
}
```

Este nodo interpola valores de coma flotante. Con esto, podríamos interpolar la intensidad de una luz, la transparencia de un objeto, o cualquier otro valor similar. También se puede "invertir" un **TimeSensor,** interpolando entre 1 en la fracción=0 y 0 en la fracción=1.

6.16.5 Creando comportamientos avanzados con el nodo Script

6.16.5.1 El nodo Script

El nodo Script es probablemente el más importante de VRML97. Convierte un lenguaje de programación simple y estático (a pesar de las animaciones) en un universo casi infinito de posibilidades. Básicamente, el nodo Script permite construir tus propios nodos con sus propios campos, eventIns y eventOuts. La sintaxis del nodo **Script** es como sigue:

```
Script
 {exposedField  MFString   url             [ ]
  field         SFBool     directOutput    FALSE
  field         SFBool     mustEvaluate    FALSE
```

También, tanto de lo siguiente como sea necesario:

```
  eventIn       Type       eventInName
  field         Type       fieldName       default value
  eventOut      Type       eventOutName
 }
```

Así como éstos, también tienen cualquier número de campos definidos por el programador, eventIns y eventOut. Se usa la palabra reservada *field*, eventIn o eventOut, según lo que se quiera hacer, y el tipo de valor que acepta el campo (por ejemplo SFBool), y finalmente el nombre. En el caso de que sea un campo, y no un eventIn o un eventOut, se debe especificar un valor para el campo. Un ejemplo de esto se muestra debajo:

```
Script {
  eventIn    SFBool   input
  field      SFBool   boolValue   TRUE
  eventOut SFBool   output
  url "filter.js"
}
```

Esto especifica un campo de cada tipo para nuestro nodo script. Un eventIn llamado *input*, un eventOut llamado *output*, y un campo llamado *boolValue* con un valor predefinido de TRUE. También especifica que el código del script se encuentra en el archivo "filter.js."

Nota: los nodos **Script** no pueden tener exposedFields.

6.16.5.2 Los Lenguajes.

Hay tres maneras de definir scripts en VRML. En primer lugar, se puede emplear código Java compilado, en un archivo .class referido con su ruta correspondiente en el campo *url*. Después, se puede usar código JavaScript en un archivo externo con extensión .js, igual que antes. Y finalmente, se puede incluir directamente el código en el campo *url*. Un ejemplo de esto último se muestra debajo:

```
url "javascript:
  function input(value, time) {
    if (value==boolValue) output = value;
  }
"
```

Un script escrito directamente se especifica usando en el nodo url "javascript:" seguido del código en ese mismo campo. Esto es apropiado para cantidades pequeñas de código, o código que se usa una única vez por archivo, donde el tiempo empleado en transmitir la información del archivo extra no compensa por escribir un archivo VRML ligeramente mayor.
Lo que sí resulta un poco más complejo es la elección de lenguaje de programación que vamos a emplear. JavaScript es un lenguaje muy simple que se aprende con facilidad, con un procesamiento simple de eventos y buena integración con VRML. Si embargo, le falta el poder y la flexibilidad de Java. Por este motivo, JavaScript es muy apropiado para scripts simples y pequeños como interruptores, filtros, y este tipo de empleos. Java es más apropiado para el trabajo duro, operaciones matemáticas complejas, o relación con el hardware. Además, al estar compilado en bytecode, aporta la ventaja de que otras personas no pueden ver el funcionamiento interno de tu programa. Si se quisiera proteger tu código VRML para que no pudiese ser visto o manipulado por otra persona, lo más apropiado sería que se utilicé Java para generar el resto del código.

6.16.5.3 Conectándolo todo.

Una vez que se haya realizado el script, se puede conectar donde se desee dentro de la cadena de eventos. Procesarán los eventos que se le envié y enviarán los eventos a los campos a los que los hayas dirigido, y los campos del nodo que se haya definido anteriormente influirán en su comportamiento interno. Los dos campos que hemos ignorado anteriormente necesitan ahora ser cubiertos con detalle. El primero es el campo *mustEvaluate.* Dice al navegador que evalué los eventos que recibe el nodo **Script** tan rápido como le sea posible. De ese modo, si se están procesando muchos datos al mismo tiempo, puede saltarse la cola de eventos y evaluarlos todos de un sólo golpe, lo cual es mucho más eficiente. Generalmente, hay que dejarlo como FALSE, a menos que compruebes que se están realizando retrasos procesando los eventos del script, en cuyo caso debes ponerlo a TRUE para evitarlo. El campo *directOutput* permite a un Script saltarse la cadena de eventos y colocar los valores de sus campos directamente en los nodos a los que tiene acceso. Esto se verá con detalle en la cuarta parte.

6.17 Sumario de la estructura de un archivo VRML.

Este archivo es un sumario de la manera en que debe componerse un escenario VRML. Muestra los nodos de agrupamiento válidos, los que están a nivel de raíz y los nodos children, junto con algo más de información útil.

6.17.1 Cabecera de un archivo VRML97

La cabecera (La primera línea) debe ser la que sigue:
#VRML V2.0 utf8
Cualquier cosa posterior a una # es interpretada como un comentario, e ignorada.

6.17.2 Nodos de agrupamiento.

Los siguientes nodos son de *agrupamiento*, lo que significa que sólo podrán contener otros nodos en su campo *children*.

- **Anchor**
- **Billboard**
- **Collision**
- **Group**
- **LOD**
- **Switch**
- **Transform**

6.17.3 Nodos Children

Los siguientes nodos son children legalmente, lo que significa que no *necesitan* estar contenidos dentro de un nodo de agrupamiento, pero que pueden estarlo. Pueden ir en la raíz del archivo, o en cualquier punto del grafo de la escena, contenidos dentro de alguno de los nodos de agrupamiento descritos arriba.

- **Anchor**
- **Background**
- **Billboard**
- **Collision**
- **ColorInterpolator**
- **CoordinateInterpolator**
- **CylinderSensor**
- **DirectionalLight**
- **Fog**
- **Group**
- **Inline**
- **LOD**
- **NavigationInfo**
- **NormalInterpolator**
- **OrientationInterpolator**
- **PlaneSensor**
- **PointLight**
- **PositionInterpolator**
- **ProximitySensor**
- **ScalarInterpolator**
- **Script**
- **Shape**

- **Sound**
- **SpotLight**
- **SphereSensor**
- **Switch**
- **TimeSensor**
- **TouchSensor**
- **Transform**
- **Viewpoint**
- **VisibilitySensor**
- **WorldInfo**
- **Child nodos redefinidos con PROTO**

6.17.4 Nodos Bindable

Los siguientes nodos son *bindable*.

- **Background**
- **Fog**
- **NavigationInfo**
- **Viewpoint**

6.17.5 Nodos Geométricos

Los siguientes nodos son *Geométricos*, y sólo pueden estar contenidos dentro del campo *geometry* de un nodo **Shape**.

- **Box**
- **Cone**
- **Cylinder**
- **ElevationGrid**
- **Extrusion**
- **IndexedFaceSet**
- **IndexedLineSet**
- **PointSet**
- **Sphere**
- **Text**

6.17.6 Nodos Especiales

Los siguientes nodos son especiales, y sólo pueden estar contenidos en determinados campos de determinados nodos.

Nodo especial	**Nodo principal**	**campo**
Appearance	Shape	appearance
AudioClip	Sound	source
Color	ElevationGrid IndexedFaceSet IndexedLineSet	color color color
Coordinate	IndexedFaceSet IndexedLineSet PointSet	coord coord coord
FontStyle	Text	fontStyle
ImageTexture	Appearance	texture
Material	Appearance	material
MovieTexture	Appearance Sound	texture source
Normal	ElevationGrid IndexedFaceSet	normal normal
PixelTexture	Appearance	texture
TextureCoordinate	ElevationGrid IndexedFaceSet	texCoord texCoord
TextureTransform	Appearance	textureTransform

Tabla 6.1 Nodos especiales.

7
METODO PARA LA GRAFICACION EN 3D

7.1 Modelado del diagrama de radiación de las antenas.

Para el modelado de los diagramas de radiación de las antenas en 3 Dimensiones, necesitaremos algo de geometría analítica y algunos nodos de VRML.
En esta sección explicaremos los pasos que ayudan a graficar dichos diagramas, en este caso tomaremos de referencia el ejemplo del dipolo de media onda ($2H/\lambda = 0.5$). El primero de los pasos es determinar la ecuación de la antena, el segundo es representar la ecuación en 3 dimensiones.
El sistema de coordenadas cartesianas a utilizar serán las de VRML. Dicho sistema de coordenadas se muestra en la figura 7.1.

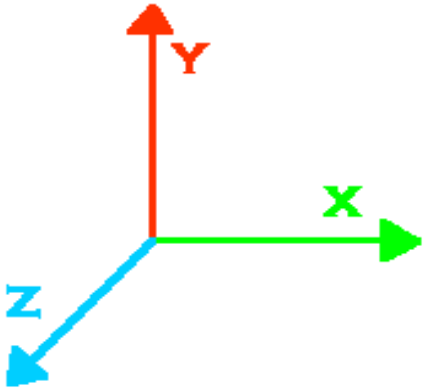


Figura 7.1 coordenadas a utilizar en VRML.

El eje X es horizontal, Y es vertical y Z atraviesa la pantalla hacia fuera.

7.1.1 Interpretación de la ecuación del campo eléctrico.

De la ecuación de campo eléctrico de la antena dipolo (Ec. 7-1) utilizamos la función de directividad (7.2), para representar el diagrama de radiación.

$$E_\theta(\theta,\phi,r,t) = \frac{j60I_m e^{-j\beta r}}{r}\left[\frac{\cos\left(\frac{2\pi H}{\lambda}\cos\theta\right)-\cos\frac{2\pi H}{\lambda}}{sen\theta}\right]sen(\varpi t-\beta r) \qquad \text{(Ec. 7-1)}$$

$$f_\theta(\theta,\phi) = \frac{\cos\left(\frac{2\pi H}{\lambda}\cos\theta\right)-\cos\frac{2\pi H}{\lambda}}{sen\theta} \qquad \text{(Ec. 7-2)}$$

Graficando la expresión 7.2 en coordenadas polares teniendo como única variable independiente θ obtenemos el diagrama de radiación del dipolo, luego el objetivo es representar dicho diagrama en 3D, esto se lograra siguiendo los siguientes pasos:

Definimos el plano θ que según la figura 7.2 estará comprendido desde θ = 0 hasta θ = π a una distancia r constante, θ = 0 esta ubicado en el eje +y, θ = π en el eje –y.

1. Se evaluara la expresión f_θ en coordenadas polares en el plano θ, como se muestra en la figura 7.3.

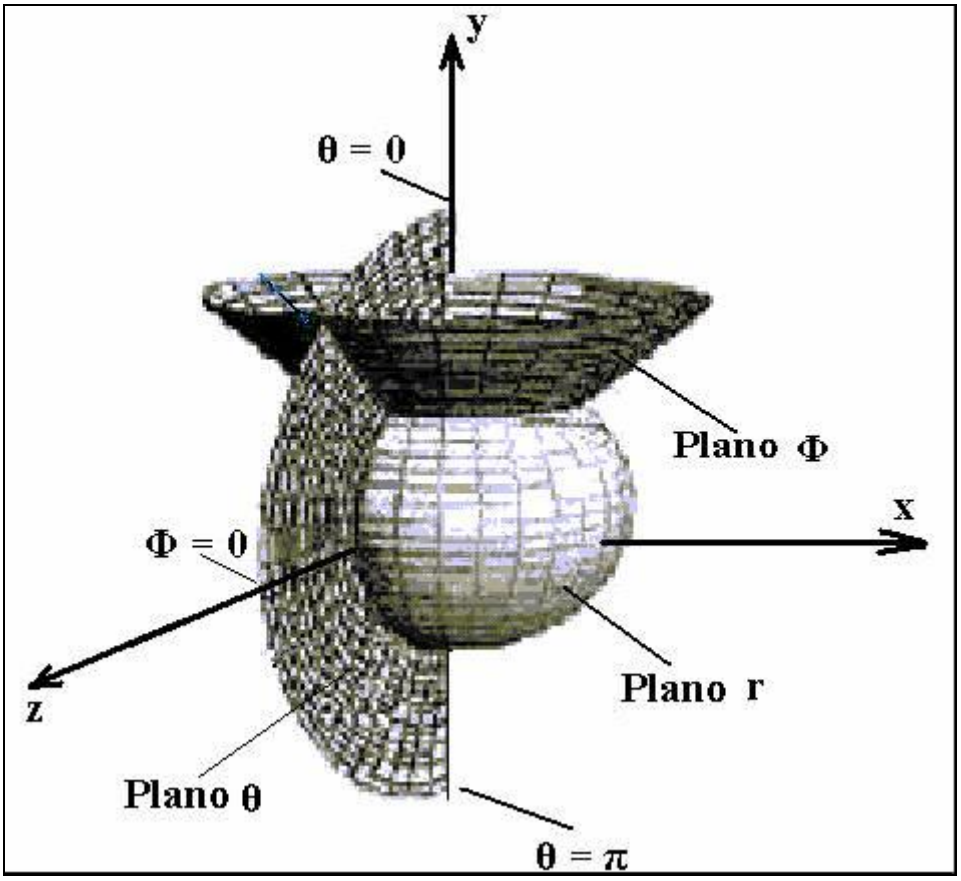


Figura 7.2 Plano θ

2. La variable Φ estará comprendido en todo el plano x-z donde Φ = 0 esta en el eje +z y Φ = π en el eje –z.

3. Empezando en $\Phi^1 = 0$, se evaluara la expresión 7.2

4. Se evaluara la expresión f_θ variando la variable θ desde 0˚ hasta 180˚ en pasos de π/resolucion2, donde resolución2 es un número entero que dependerá de la resolución que se quiera del gráfico. para obtener el diagrama polar según se muestra en la figura 7.3.

[1] En este caso la expresión del dipolo f_θ no tiene variable Φ pero habrán expresiones que si dependerán de dicha variable como en el caso del dipolo con interferencia que se vera mas adelante.

5. Se incrementa la variable Φ en pasos de 2π/resolucion1 y luego se continua con paso 5 hasta que Φ = 2 π.

6. Se incrementa la variable Φ en pasos de 2π/resolucion1 y luego se continua con paso 5 hasta que Φ = 2 π.

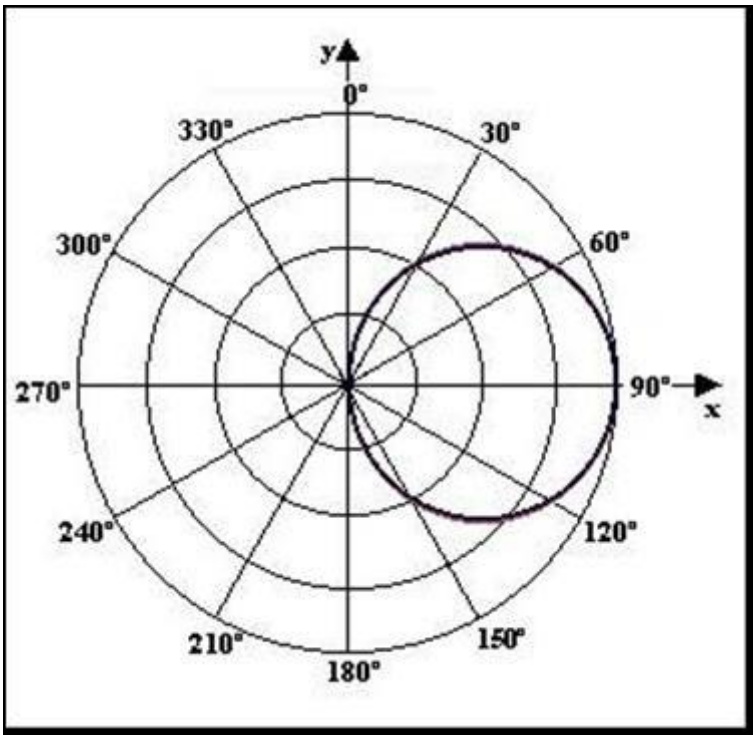


Figura 7.3. Representación del diagrama de radiación en coordinas polares.

7.1.2 VRML y Visual Basic como apoyo en la graficación en 3D.

En la sección 6 se vio aspectos generales sobre como crear mundos 3D en VRML, pero en unos de los objetivos de nuestro proyecto necesitamos un entorno en el cual nosotros podamos cambiar variables para observar como se comporta el diagrama de radiación en 3D, característica que VRML no la posee.
La herramienta más idónea que utilizamos fue Visual Basic por su fácil[2] programación, entorno amigable y lo más importante por poseer la herramienta *WebBrowser*, que es simplemente un visor de Internet Explorer.
Esta herramienta nos será de gran utilidad recordando que los archivos VRML solo los podemos observar a través de Internet Explorer.

7.1.2.1 El nodo IndexedFaceSet.

El principal nodo que utilizaremos para la representación del diagrama de radiación en 3D es el nodo IndexedFaceSet, que se vio en la sección 6.11.1, recordando la sintaxis del nodo IndexedFaceSet es:

[2] Esto es relativo dependiendo del tipo de programador que sea uno.

```
IndexedFaceSet
   {SFNode     coord          NULL
    MFInt32    coordIndex     [ ]
    SFBool     solid          TRUE
   }
```

Con este nodo crearemos un área superficial diferencial de cuatro vértices, dicha área será coplanar pero debido a su ínfimo tamaño formara áreas superficiales (diagrama de radiación) de diversas formas.
Primero se necesitara definir las coordenadas de los vértices en el campo **coord**, luego se definirá que vértices formaran un diferencial de área superficial especifico en el campo **coordIndex**.

Por ejemplo si queremos definir un cuadrado con las coordenadas p(x y z):

P0(-1 1 0), P1(1 1 0), P2(1 -1 0), P3(-1 -1 0)

El código lo definiríamos de la siguiente forma:

```
Shape  {appearance Appearance
              {  material Material
                   {diffuseColor 0.5 1 0.5
                   }
              }
        geometry IndexedFaceSet
              {coord Coordinate
                 {point [ -1 1 0, 1 1 0, 1 -1 0, -1 -1 0
                        ]
              }
              coordIndex [ 0 3 2 -1
                           ]
              solid FALSE
              }
       }
```

La figura se observaría como en la figura 7.4 a).
Pero si utilizáramos los vértices P3, P1, P0 se definiría el campo coordIndex [3 1 0 -1] y se observaría como en la figura 7.4 b).
Si utilizáramos los vértices P3, P2, P1 se definiría el campo coordIndex [3 2 1 -1] la forma se muestra en la figura 7.4 c).
El campo solid se le adjudica el valor de FALSE para que se pueda ver el revés de la cara.

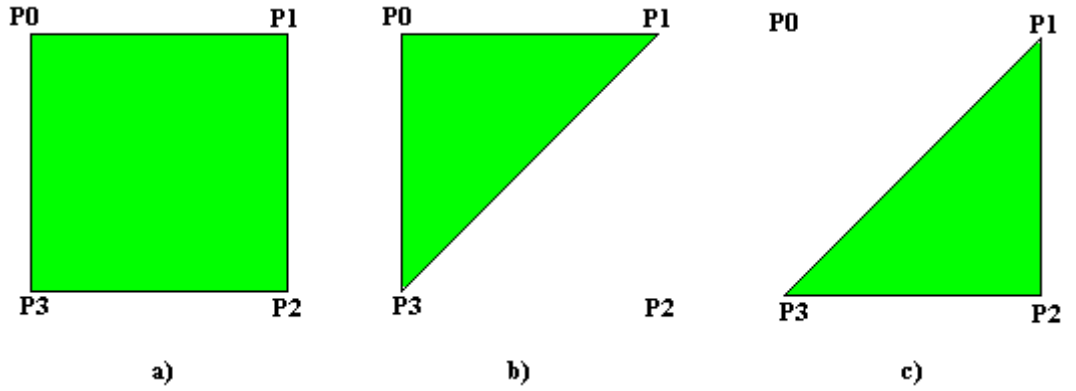


Figura 7.4. Representación de una figura variando valores del campo coordIndex.

7.1.2.2 La herramienta WebBrowser de Visual Basic.

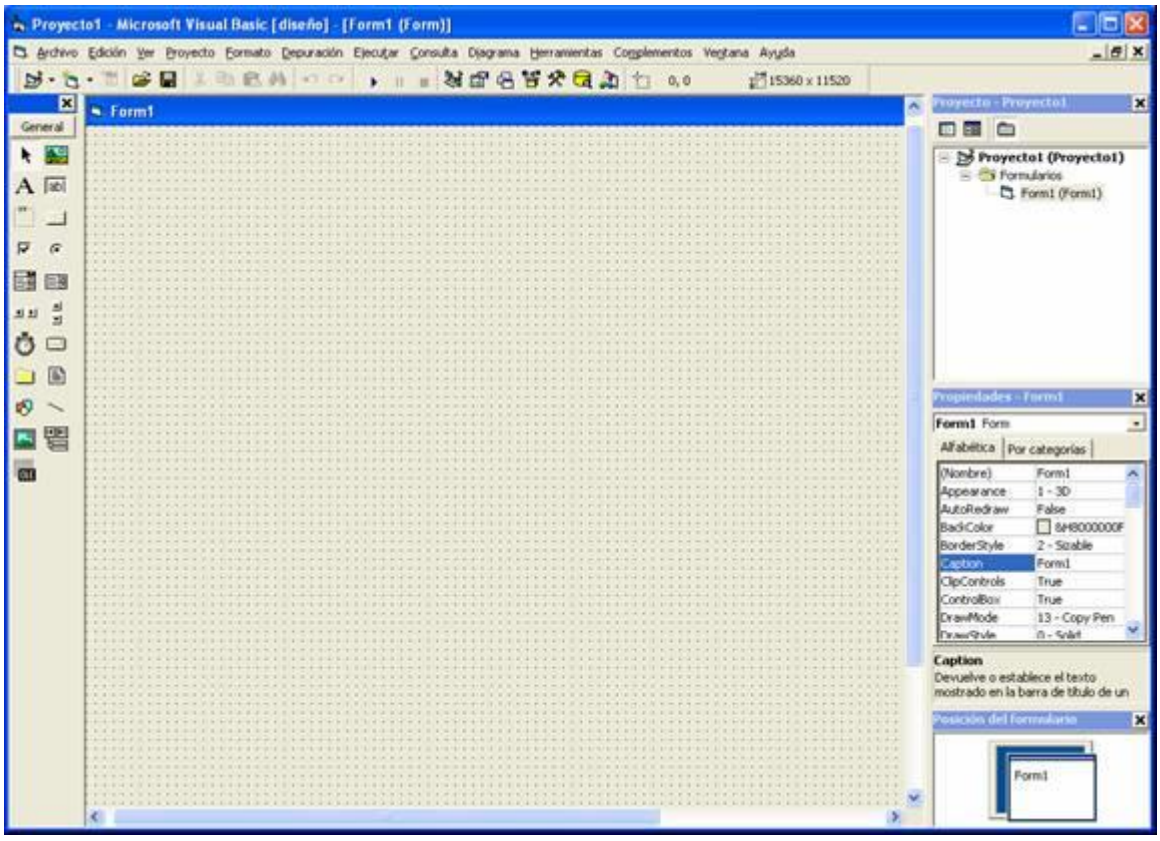


Figura 7.5. Presentación principal de Visual Basic.

Podemos crear figuras geométricas avanzadas en VRML con el nodo **IndexedFaceSet,** pero debido a que los coordenadas de los vértices de las áreas diferenciales dependerán de cómo se comporte la expresión $f_{\theta,}$ necesitamos un lenguaje de programación que nos vaya determinando las coordenadas de los vértices y al mismo tiempo que nos muestre el diagrama en 3D a través de Internet Explorer.
La solución la tenemos con Visual Basic por su fácil programación y por poseer la herramienta WebBrowser.

WebBrowser es un visor de Internet Explorer, que se le programa la dirección del archivo o pagina web que se quiere observar.
Para agregar el WebBrowser hay que hacer clip derecho en el cuadro de herramientas extremo izquierdo de la figura 7.5, aparecerá un menú luego hacer clip izquierdo en componentes y aparecerá el cuadro de la figura 7.6, poner cheque a Microsoft Internet Control y ya tendremos instalada la herramienta.

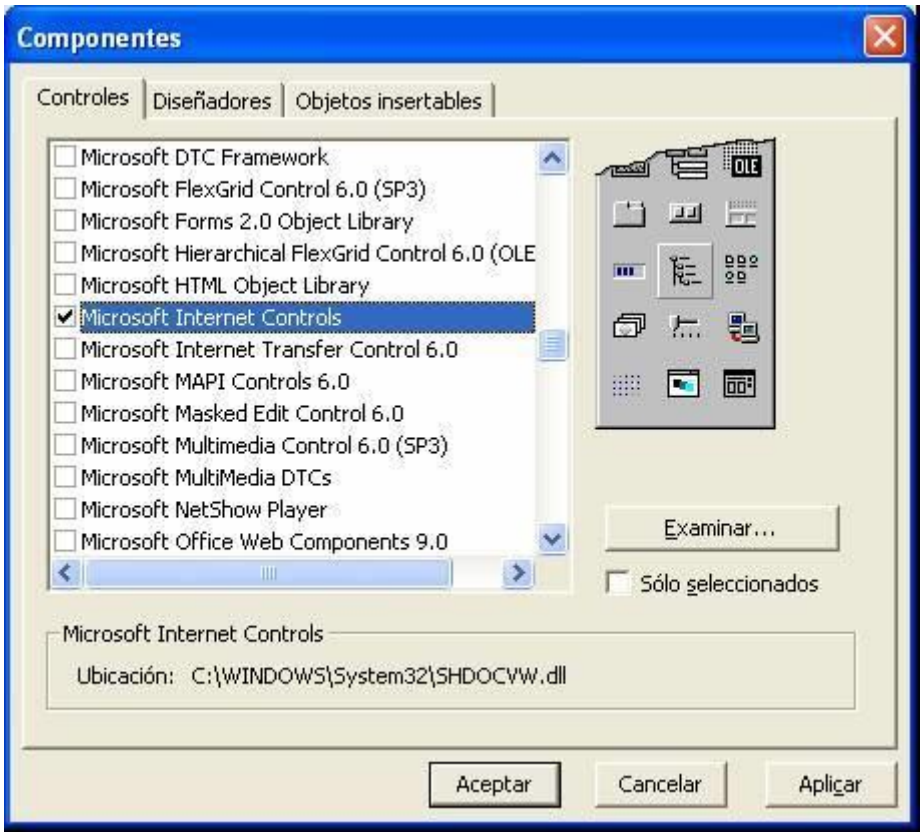


Figura 7.6. Componentes del cuadro de herramientas.

7.3 Ejecución del programa.

Cuando se empiece a correr el programa en Visual Basic, aparecerá la ventana principal que se observa en la figura 7.7, donde aparecen cinco opciones:

1. Dipolo elemental sin interferencia.
2. Dipolo elemental conm interferencia.
3. Antena Yagi.
4. Antena Parabólica.
5. salir del programa.

Figura 7.7. Ventana principal del programa.

Si queremos ver el patrón de radiación de la antena dipolo se visualizara la siguiente pantalla.

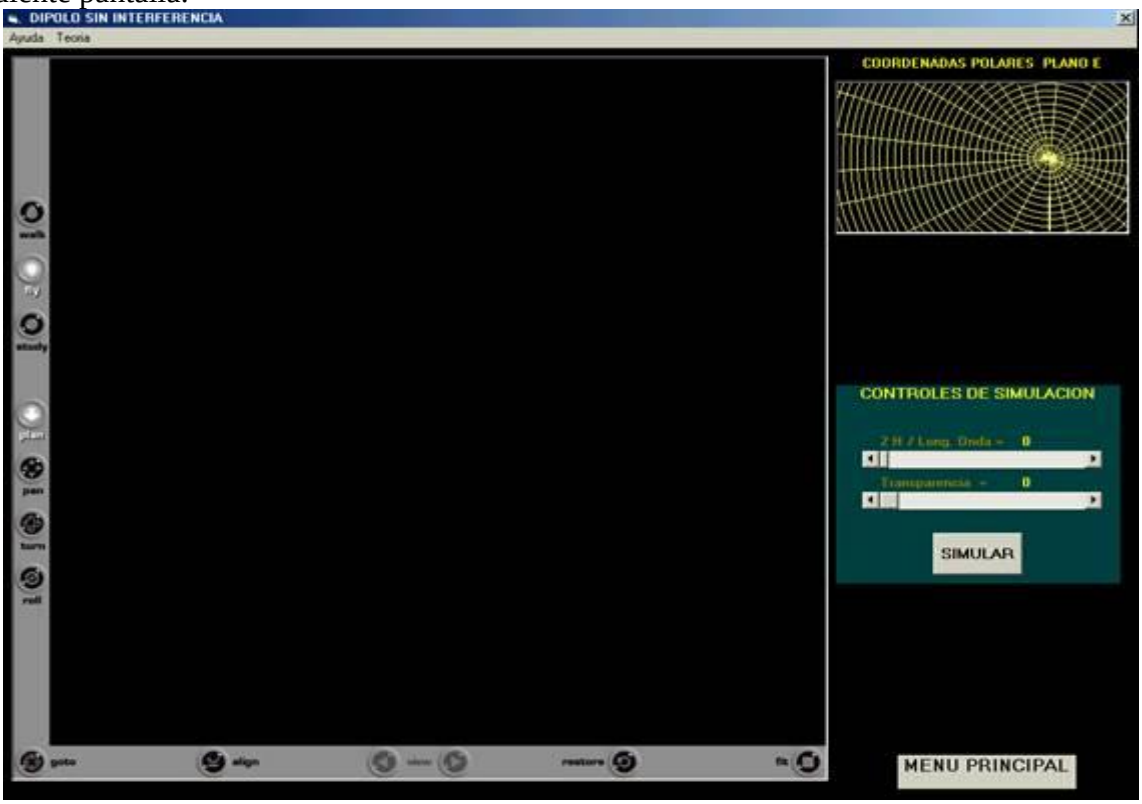


Figura 7.8. Ventana donde se muestra diagrama.

En la pantalla de la figura 7.8 aparece la información de las características de la antena dipolo, también se podrán variar parámetros como la relación 2H/λ y el nivel de transparencia del patrón de radiación de la antena.
Luego cuando ya hayamos elegido las opciones y le demos clic al botón "Simular patrón de radiación sin interferencia" y el diagrama de radiación de la antena dipolo será simulado en la ventana inferior derecha (parte donde se encuentra la figura de VRML).

En la figura 7.9 se muestra el patrón de radiación de la antena dipolo, cuando se le aplica una relación de onda de 0.5 y una transparencia de 0.5.

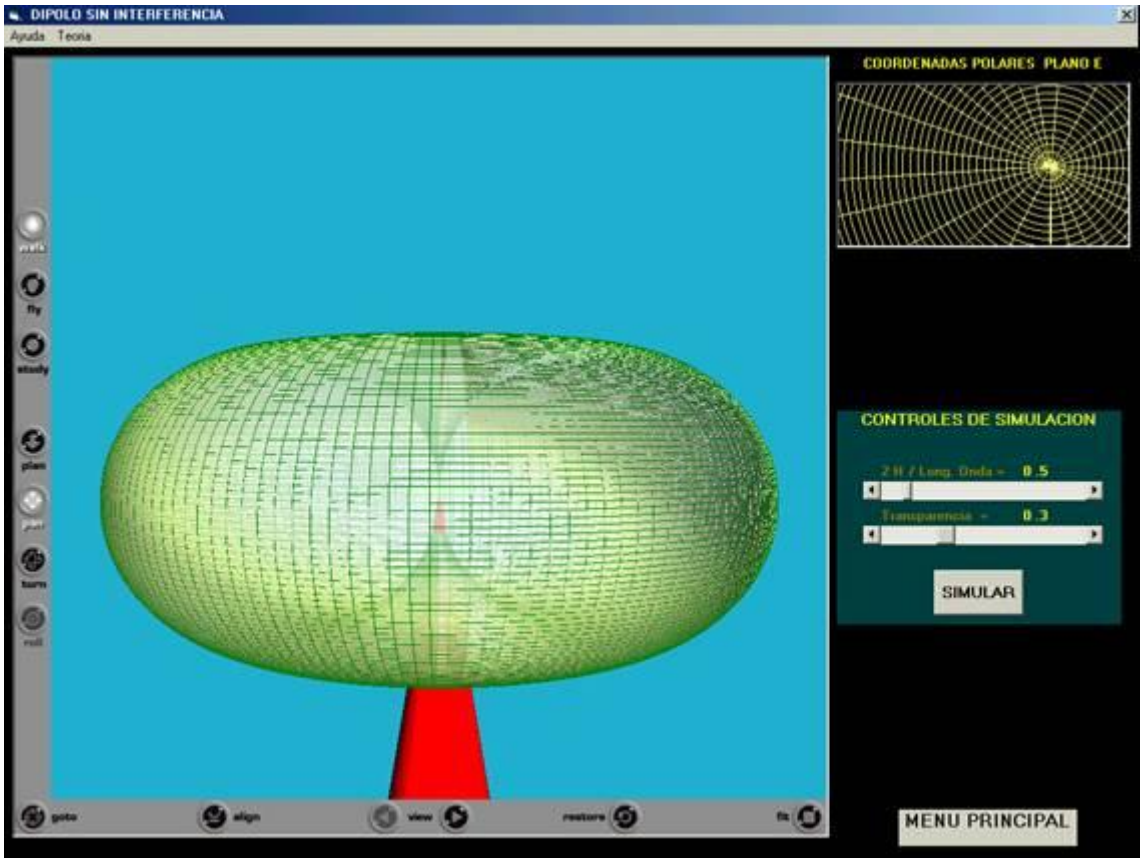


Figura 7.9. Diagrama del dipolo con relación 2H/λ = 0.5 y transparencia de 0.3.

Se observa en la figura 7.9 como se grafica el patrón de radiación en dos dimensiones (Ventana superior derecha) para el dipolo y también el diagrama en tres dimensiones (Ventana inferior derecha) cuando se le aplica una relación de onda de 0.5 y una transparencia de 0.5.
En la figura 7.10 y 7.11 se dan unos ejemplos de la forma en que cambia el patrón de radiación del Dipolo cuando se le aplica una relación de onda diferente.

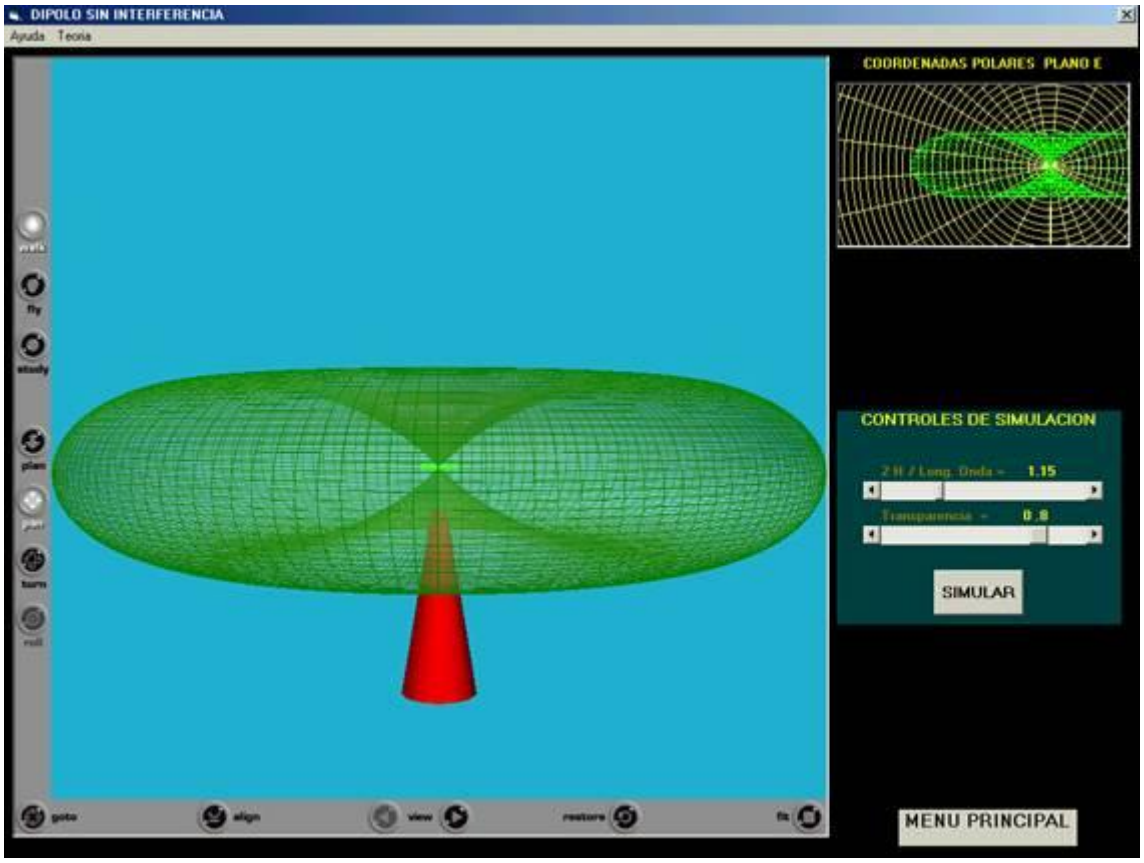


Figura 7.10. Diagrama del dipolo con relación 2H/λ = 1.15 y transparencia de 0.8.

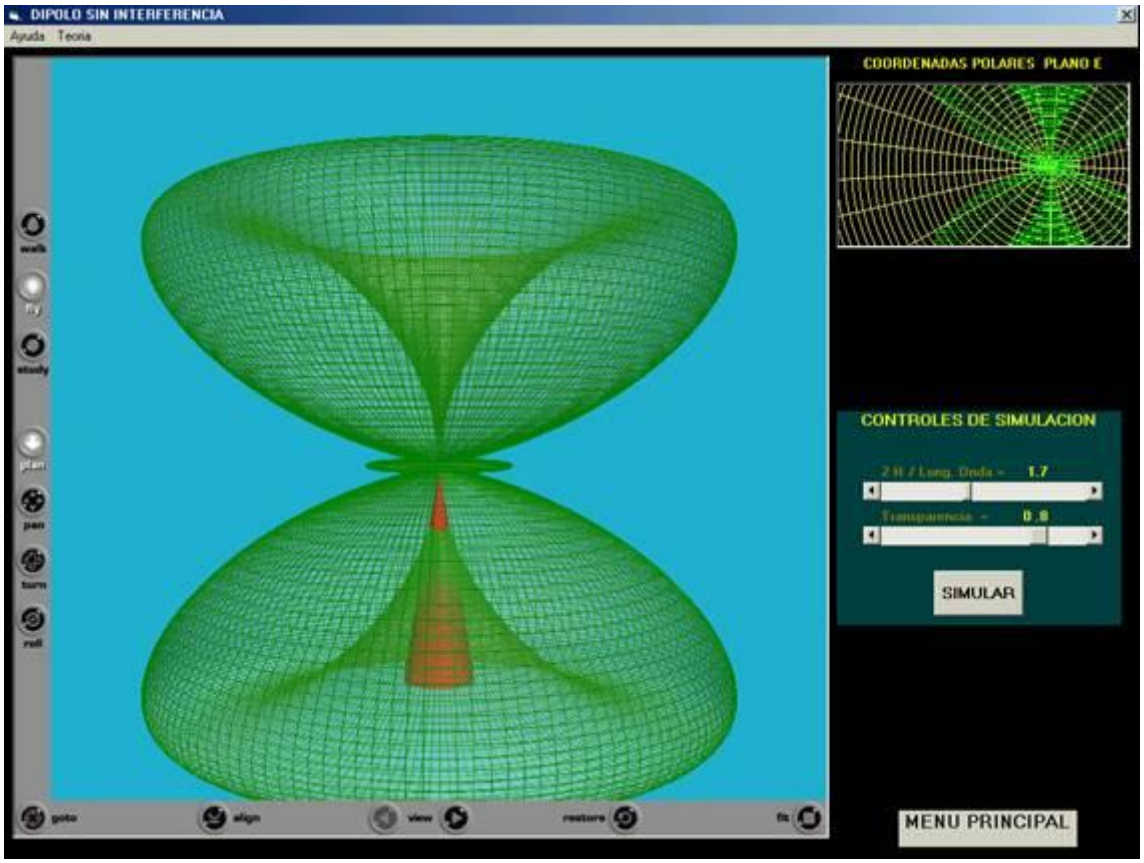


Figura 7.11. Diagrama del dipolo con relación 2H/λ = 1.7 y transparencia de 0.8.

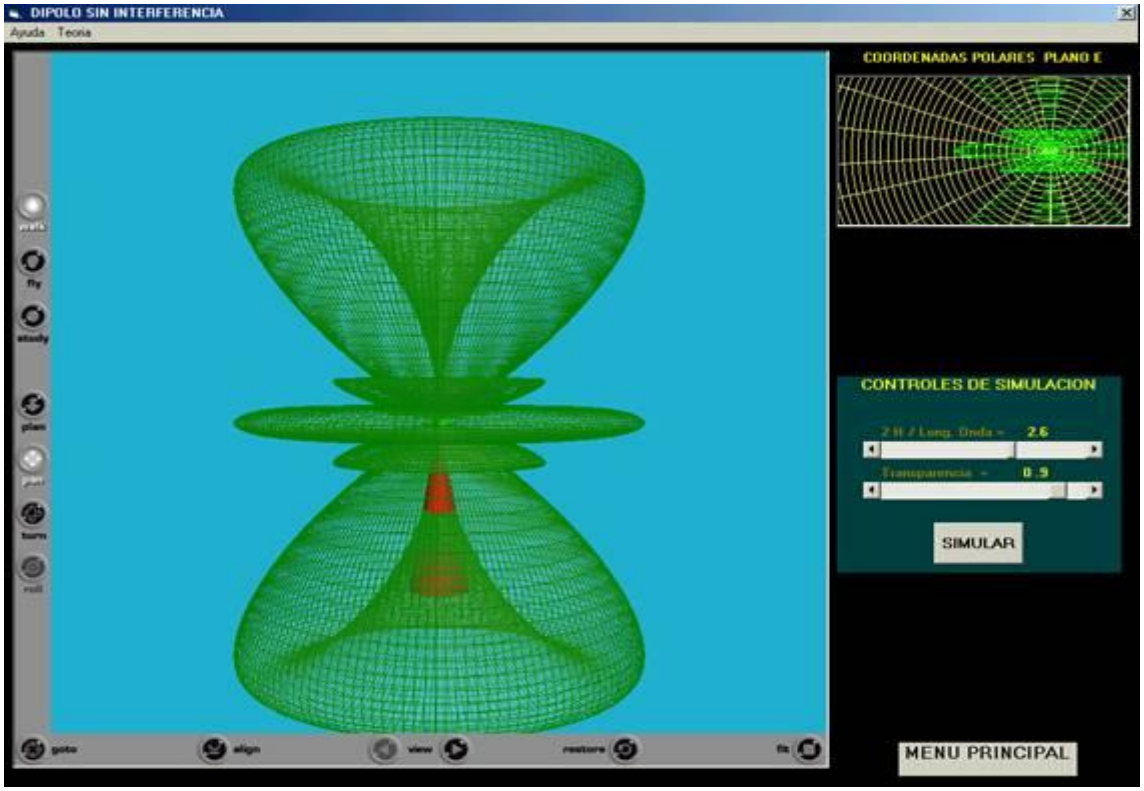


Figura 7.12 diagrama del dipolo con relación 2H/λ = 2.6 y transparencia de 0.9.

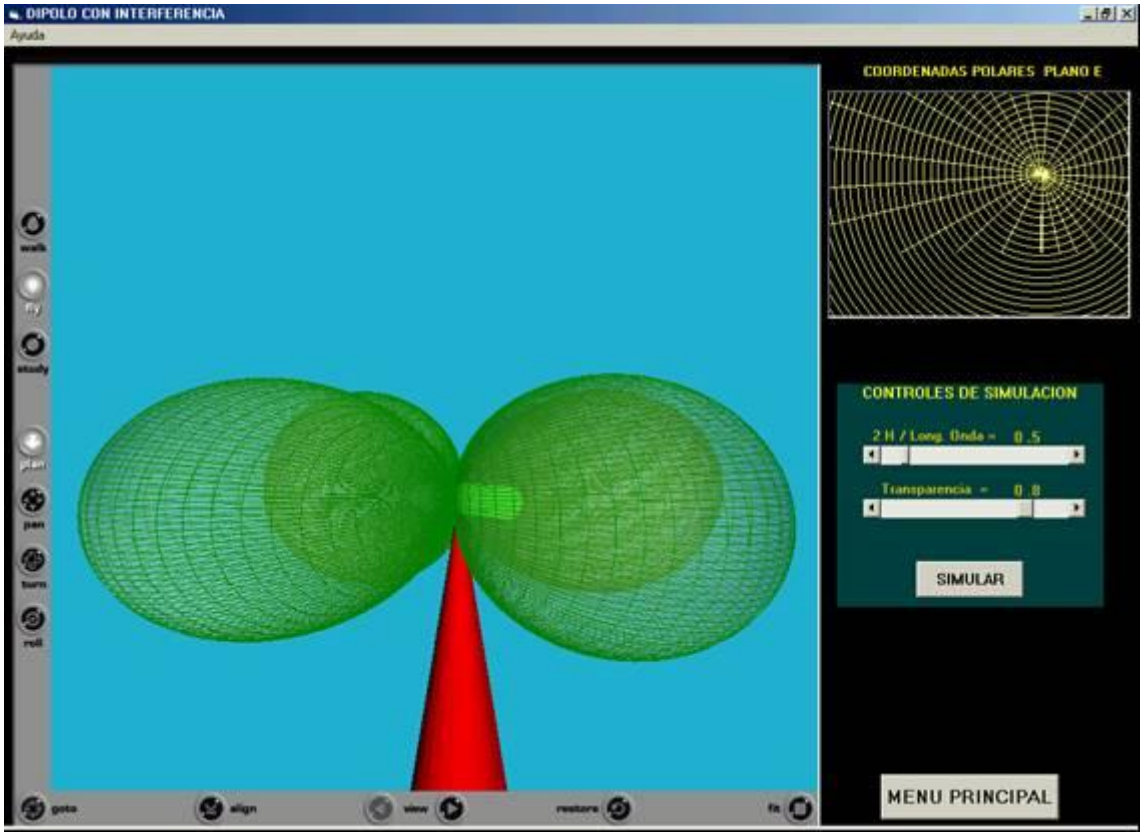


Figura 7.15. Diagrama del dipolo con relación 2H/λ = 0.5 con interferencia y transparencia de 0.8.

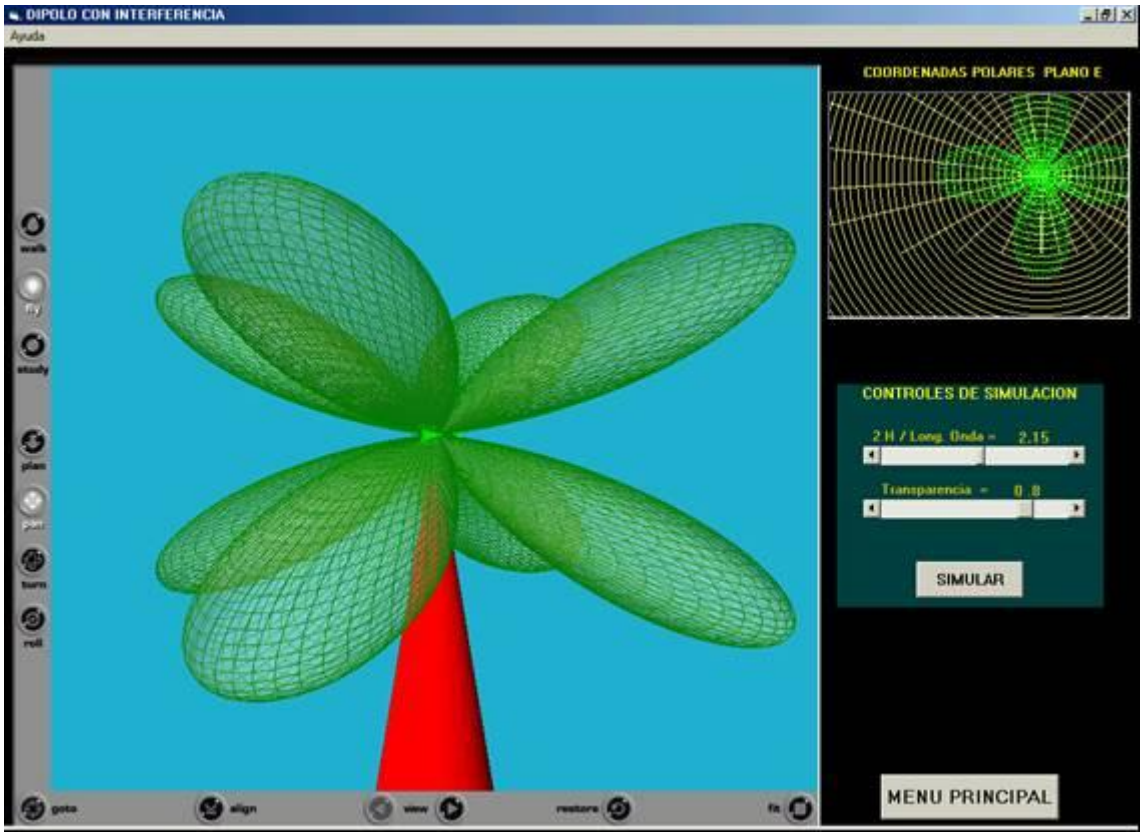


Figura 7.16. Diagrama del dipolo con relación 2H/λ = 2.15 con interferencia y transparencia de 0.9.

APÉNDICE A

Deducción de las ecuaciones del diagrama de directividad de un dipolo elemental [3]

El primer paso es paso es de obtener el potencial vector **A** en P. La expresión general de **A** esta dada por

$$A(r) = \frac{\mu}{4\pi}\int \frac{J(t-\frac{r}{v})}{R}dV \qquad \text{(Ec. A-1)}$$

La integración en el volumen en (A-1) consiste en una integración sobre el área transversal del alambre y una integración a lo largo de su longitud. La densidad de corriente **J**, integrada en el área transversal del alambre es precisamente la corriente I, y como se supone es constante a lo largo de su longitud, la integración en la longitud da IDB. Por tanto la expresión del potencial vector resulta

$$A_z = \frac{\mu}{4\pi}\frac{Idl\cos\varpi(t-\frac{r}{v})}{r} \qquad \text{(Ec. A-2)}$$

El potencial vector tiene la misma dirección que el elemento de corriente, en este caso la dirección z, y esta retardado en r/v segundos. La intensidad de campo magnético **H** se obtiene con la relación

$$\mu H = \nabla x A \qquad \text{(Ec. A-3)}$$

Se tienen las componentes de **H** en función de A_r, A_θ, A_ϕ,

$$A_r = A_z\cos\theta \qquad A_\theta = -A_z sen\theta \qquad A_\phi = 0 \qquad \text{(Ec. A-4)}$$

Entonces de las expresiones de las componentes esféricas del rotacional y notando que debido a la simetría $\partial/\partial\phi = 0$,

$$\mu H_r = (\nabla x A)_r = 0$$
$$\mu H_\theta = (\nabla x A)_\theta = 0$$

$$H_\phi = \frac{1}{\mu r}\left[\frac{\partial}{\partial r}(rA_\theta) - \frac{\partial A_r}{\partial\theta}\right]$$

$$H_\phi = \frac{Idl}{4\pi r}\left\{\frac{\partial}{\partial r}\left[-sen\theta\cos\varpi(t-\frac{r}{v})\right] - \frac{\partial}{\partial\theta}\left[\frac{\cos\theta}{r}\cos\varpi(t-\frac{r}{v})\right]\right\}$$

$$H_\phi = \frac{Idlsen\theta}{4\pi}\left[\frac{-\varpi.sen\,\varpi(t-\frac{r}{v})}{rv} + \frac{\cos\varpi(t-\frac{r}{v})}{r^2}\right] \qquad \text{(Ec.A-5)}$$

La intensidad de campo eléctrico **E** puede obtenerse de **H** por medio de la primera ecuación de Maxwell, que en el punto P (en el espacio libre) es:

$$\nabla \times H = \varepsilon \dot{E}$$

$$E = \frac{1}{\varepsilon}\int \nabla \times H dt \qquad \text{(Ec. A-6)}$$

Tomando el rotacional de (A-5) e integrando respecto al tiempo (no importa el orden) se obtiene para las componentes de **E**

$$E_\theta = \frac{Id\ell sen\theta}{4\pi\varepsilon}\left(\frac{-\varpi.sen\varpi t'}{r\nu^2} + \frac{\cos\varpi t'}{r^2\nu} + \frac{sen\varpi t'}{\varpi r^3}\right) \qquad \text{(Ec. A-7)}$$

$$E_r = \frac{2Id\ell\cos\theta}{4\pi\varepsilon}\left(\frac{\cos\varpi t'}{r^2\nu} + \frac{sen\varpi t'}{\varpi r^3}\right) \qquad \text{(Ec. A-8)}$$

Y volviendo a escribir (A-5)

$$H_\phi = \frac{Id\ell sen\theta}{4\pi}\left(\frac{-\varpi.sen\varpi t'}{r\nu} + \frac{\cos\varpi t'}{r^2}\right) \qquad \text{(Ec. A-9)}$$

En donde t'=(t - r/υ).

Es un tanto sorprendente encontrar que algo tan aparentemente simple como un elemento de corrientes de lugar a un campo electromagnético tan complicado como el dado por (A-7), (A-8) y (A-9).

Potencia radiada por un elemento de corriente.

El flujo de potencia por unidad de área en el punto P estará dado por el vector de poynting en tal punto. El vector de Poynting instantáneo esta dado por $\tilde{E}\times\tilde{H}$ y tendrá componentes θ y r. Sustituyendo υ por c ≈ 3×10^8 para una propagación en el espacio libre, la componente θ del vector instantáneo de Poynting será

$$P_\theta = -E_r H_\phi$$

$$P_\theta = \frac{I^2 d\ell^2 sen2\theta}{16\pi^2\varepsilon}\left(\frac{sen^2\varpi t'}{r^4 c} - \frac{\cos^2\varpi t'}{r^4 c} - \frac{sen\varpi t'\cos\varpi t'}{\varpi r^5} + \frac{\varpi sen\varpi t'\cos\varpi t'}{r^3 c^2}\right)$$

$$P_\theta = \frac{I^2 d\ell^2 sen2\theta}{16\pi^2\varepsilon}\left(\frac{-\cos 2\varpi t'}{r^4 c} - \frac{sen2\varpi t'}{2\varpi r^5} + \frac{\varpi sen2\varpi t'}{2r^3 c^2}\right) \qquad \text{(Ec. A.10)}$$

El valor medio de sen2ωt'y cos2ωt' en un ciclo completo es cero. Por tanto, para todo valor de r, el valor medio de P_θ en un ciclo completo es cero. P_θ representa solamente unos flujos transitorios de potencia que van y vienen en la dirección θ sin producir flujo neto medio de potencia.
El vector radial de Poynting esta dado por

$$P_r = E_\theta H_\phi$$

$$P_r = \frac{I^2 d\ell^2 sen^2\theta}{16\pi^2\varepsilon}\left(\frac{sen\varpi t'\cos\varpi t'}{\varpi r^5} + \frac{\cos^2\varpi t'}{r^4 c} - \frac{\varpi sen\varpi t'\cos\varpi t'}{r^3 c^2} - \frac{sen^2\varpi t'}{r^4 c} - \frac{\varpi sen\varpi t'\cos\varpi t'}{r^3 c^2} + \frac{\varpi^2 sen\varpi t'}{r^2 c^3}\right)$$

$$P_r = \frac{I^2 d\ell^2 sen^2\theta}{16\pi^2\varepsilon}\left(\frac{sen2\varpi t'}{2\varpi r^5} + \frac{\cos 2\varpi t'}{r^4 c} - \frac{\varpi sen2\varpi t'}{r^3 c^2} + \frac{\varpi^2(1-\cos 2\varpi t')}{2r^2 c^3}\right) \qquad \text{(Ec. A-11)}$$

El valor medio del vector radial de Poynting en un ciclo se deberá a parte del último término y será

$$P_{r(av)} = \frac{\varpi^2 I^2 d\ell^2 sen^2\theta}{32\pi^2 r^2 c^3 \varepsilon}$$

$$P_{r(av)} = \frac{\eta}{2}\left(\frac{\varpi I d\ell sen\theta}{4\pi r c\varepsilon}\right)^2 \text{ W/m}^2 \qquad \text{(Ec. A-12)}$$

Ninguno de los términos de las expresiones de los vectores de Poynting representa un flujo medio de potencia, salvo el de la expresión (A-12). ***Los únicos términos de E y H que contribuyen a este flujo medio de potencia son los términos de radiación o inversos a la distancia***. A gran distancia de la fuente estos términos de radiación son los únicos que tienen valores apreciables, pero incluso en las proximidades del elemento de corriente, en donde predominan los campos de inducción y eléctrico, solo los términos en 1/r contribuyen a un flujo saliente medio de potencia. La densidad de potencia es proporcional al producto de E y H (o bien E^2 o H^2), de manera que E y H, que contribuyen al flujo medio radial de potencia, debe ser proporcional a 1/r. Las componentes de E y H que sean inversamente proporcionales a r^2 o r^3 pueden contribuir a un flujo transitorio de energía en la región entre capas, pero esta energía debe volver a la fuente, pues no puede ser almacenada permanentemente en un volumen finito del medio.

$$E_\theta = \frac{\varpi I d\ell sen\theta}{4\pi\varepsilon v^2 r}$$

$$E_\theta = \frac{\eta I d\ell sen\theta}{2\lambda r}$$

$$E_\theta = \frac{60\pi I d\ell sen\theta}{\lambda r} \qquad \text{(Ec. A-13)}$$

Según las ecuaciones (A-7), (A-8) y (A-9), las amplitudes de los campos de radiación de un elemento de corriente Id ℓ son

$$H_\phi = \frac{\varpi I d\ell sen\theta}{4\pi v r}$$

$$H_\phi = \frac{I d\ell sen\theta}{2\lambda r} \qquad \text{(Ec. A-14)}$$

Los términos de radiación de E_θ y H_ϕ están en fase y relacionados por $\frac{E_\theta}{H\phi} = \eta$

APÉNDICE B

Deducción de las ecuaciones del diagrama de directividad de un dipolo de longitud variable.

Se supondrá que la corriente esta distribuida sinosoidalmente,

$$I = I_m sen\beta(h - z) \qquad z>0$$

$$I = I_m sen\beta(h + z) \qquad z<0$$

En donde I_m es el valor de la corriente en el vientre o corriente máxima. La expresión del potencial vector en un punto P debido al elemento de corriente Idz será

$$dA_z = \frac{\mu I e^{-j\beta R} dz}{4\pi R}$$

Donde R es la distancia desde el elemento de corriente al punto P. El potencial vector total debido a todos los elementos de corriente será

$$A_z = \frac{\mu}{4\pi}\int_{-H}^{0} \frac{I_m sen\beta(h+z)e^{-j\beta R}dz}{R} + \frac{\mu}{4\pi}\int_{0}^{H} \frac{I_m sen\beta(h-z)e^{-j\beta R}dz}{R} \qquad \text{(Ec. B-1)}$$

Como solo se necesitan los campos alejados o de radiación en este problema es posible hacer aproximaciones simplificadoras. Para el factor inverso de la distancia (la R en el denominador) será valido escribir

$$R \approx r$$

Sin embargo, para la R en el factor de fase en el numerador es la diferencia entre R y r, lo que es importante. Para grandes valores de R las líneas al punto P son esencialmente paralelas, pudiéndose escribir para la R en el factor de fase y de modo aproximado

$$R = r - z\cos\theta$$

Así la expresión de A_z queda de la forma

$$A_z = \frac{\mu I_m e^{-j\beta r}}{4\pi}\left[\int_{-H}^{0} sen\beta(h+z)e^{j\beta z\cos\theta}dz + \frac{\mu}{4\pi}\int_{0}^{H} sen\beta(h-z)e^{j\beta z\cos\theta}dz\right] \qquad \text{(Ec. B-2)}$$

$$A_z = \frac{\mu I_m e^{-j\beta r}}{4\pi}\left[\int_{-H}^{0} sen\beta(h+z)\cos(\beta z\cos\theta)dz + \int_{0}^{H} sen\beta(h-z)\cos(\beta z\cos\theta)dz\right]$$

$$A_z = \frac{\mu I_m e^{-j\beta r}}{4\pi}\left[\frac{1}{2}\left\langle\int_{0}^{H}\{sen[\beta(h+z)+\beta z\cos\theta]+sen[\beta(h+z)-\beta z\cos\theta]\}dz\right\rangle + \right.$$

$$\left.\frac{1}{2}\left\langle\int_{0}^{H}\{sen[\beta(h-z)+\beta z\cos\theta]+sen[\beta(h-z)-\beta z\cos\theta]\}dz\right\rangle\right]$$

L.

$$A_z = \frac{\mu I_m e^{-j\beta r}}{2\pi\beta r}\left[\frac{\cos(\beta H\cos\theta)-\cos\beta H}{sen^2\theta}\right] \qquad \text{(Ec. B-3)}$$

Recordamos del problema la que cuando la corriente esta enteramente en la dirección z.

$$\mu H_{\phi} = -\frac{\partial A_z}{\partial r} sen\theta$$

La expresión de la intensidad del campo magnético en punto distante será

$$H_{\phi} = \frac{jI_m e^{-j\beta r}}{2\pi r}\left[\frac{\cos(\beta H\cos\theta) - \cos\beta H}{sen\,\theta}\right] \qquad \text{(Ec. B-4)}$$

De donde solo se ha retenido él termino inverso a la distancia. La intensidad de campo eléctrico de radiación será

$$E_\theta = \eta H_{\phi}$$

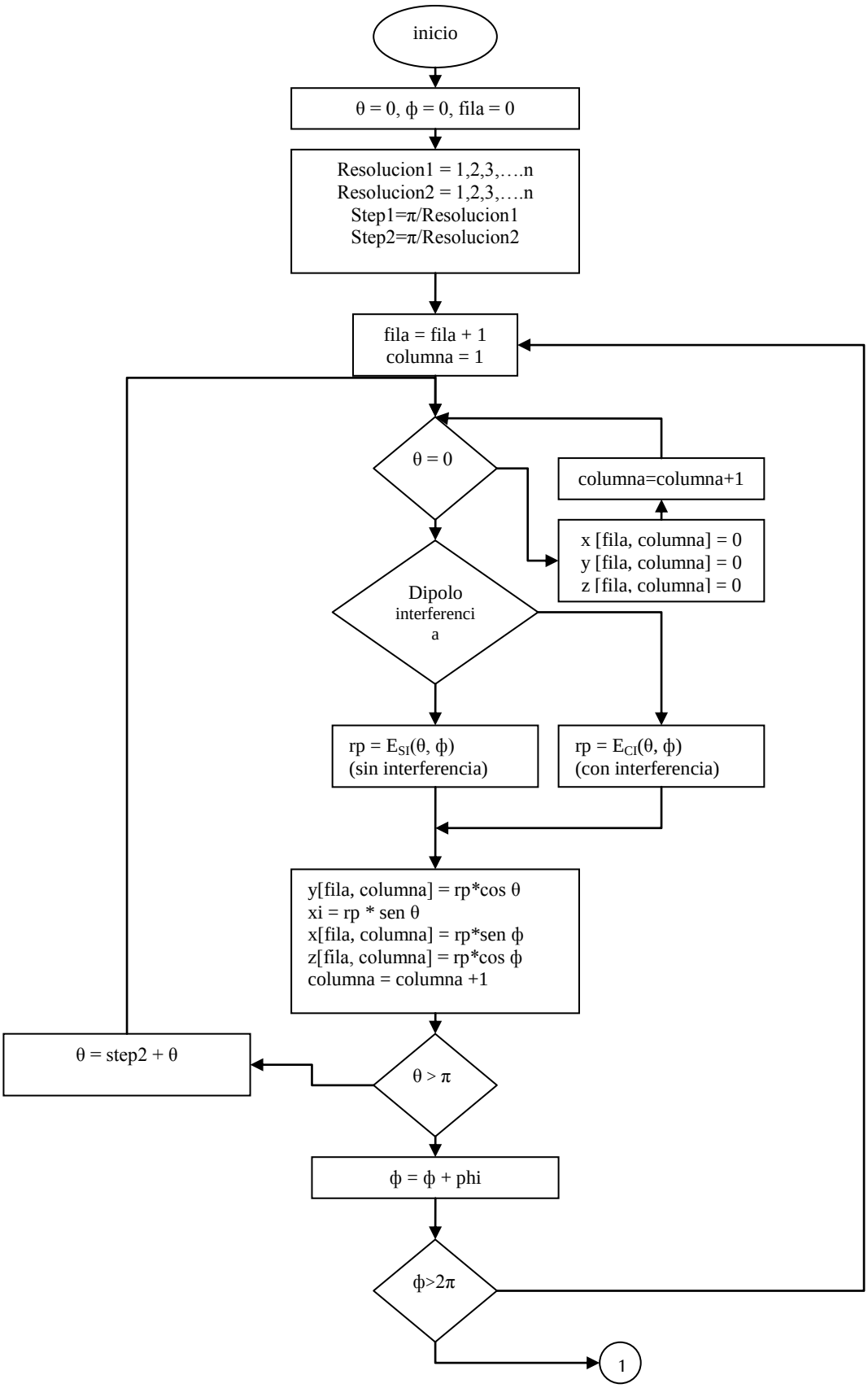
$$E_{\theta} = \frac{j60I_m e^{-j\beta r}}{r}\left[\frac{\cos(\beta H\cos\theta) - \cos\beta H}{sen\,\theta}\right] \qquad \text{(Ec. B-5)}$$

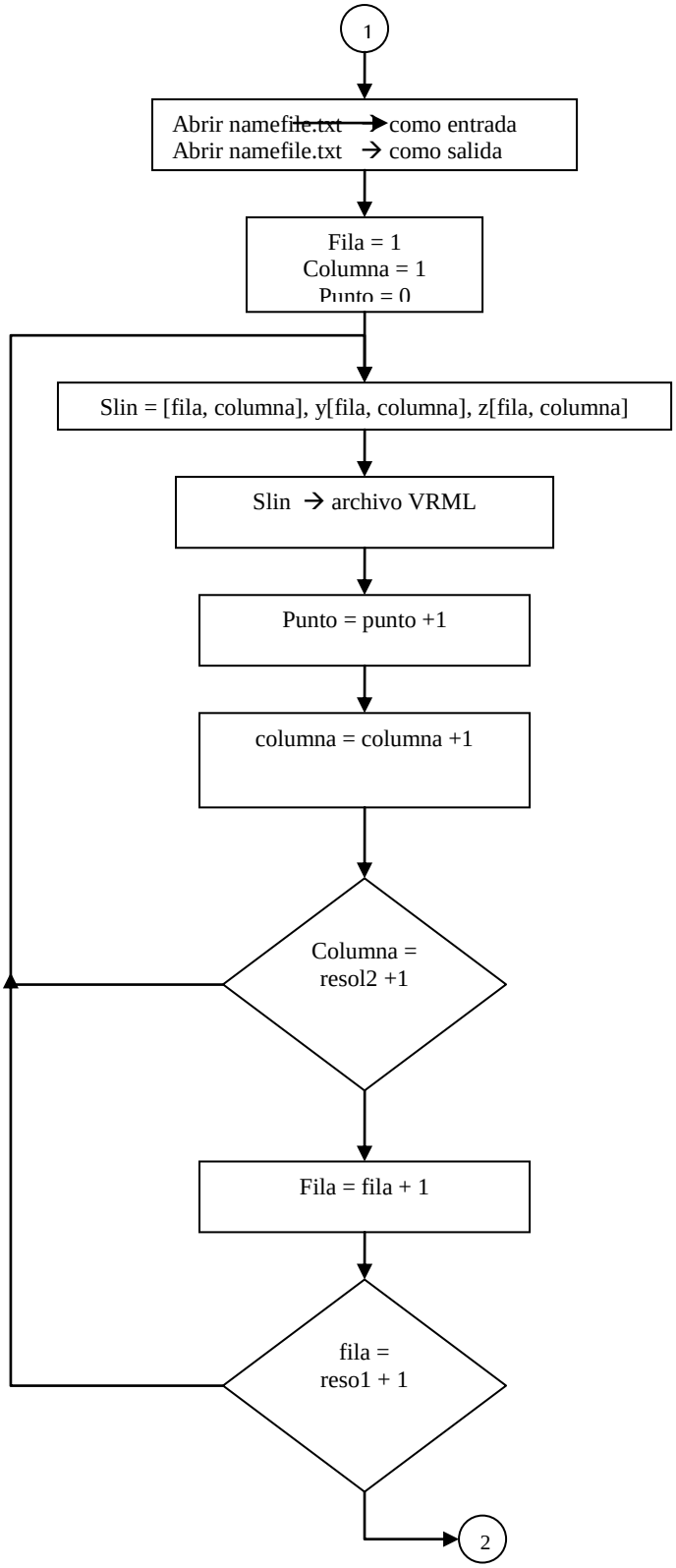
$$E_{\theta} = \frac{j60I_m e^{-j\beta r}}{r}\left[\frac{\cos\left(\frac{2\pi H}{\lambda}\cos\theta\right) - \cos\frac{2\pi H}{\lambda}}{sen\,\theta}\right] \qquad \text{(Ec. B-6)}$$

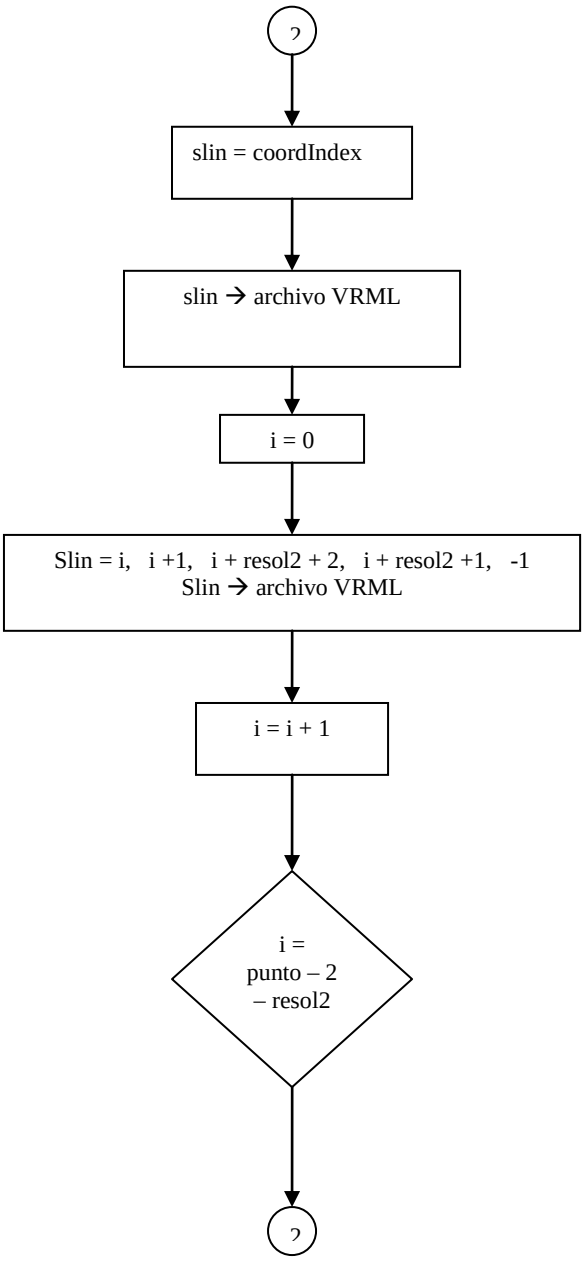
.

APÉNDICE C

Flujograma y programa en VRML y Visual Basic.




1
Abrir namefile.txt como entrada
Abrir namefile.txt → como salida
Fila = 1
Columna = 1
Punto = 0
Slin = [fila, columna], y[fila, columna], z[fila, columna]
Slin → archivo VRML
Punto = punto +1
columna = columna +1
Columna = resol2 +1
Fila = fila + 1
fila = reso1 + 1
2


2
slin = coordIndex
slin → archivo VRML
i = 0
Slin = i, i +1, i + resol2 + 2, i + resol2 +1, -1
Slin → archivo VRML
i = i + 1
i =
punto – 2
– resol2
2

Programa en VRML.

```
#VRML V2.0 utf8

   #####    TIMER     ###
DEF TIMER TimeSensor
   {cycleInterval 30
    loop TRUE
   }

  ####  CAMARA GIRATORIA  ####
DEF ORITGT Transform
{children
 [Transform
   {translation 0 0 30
    children
   [ DEF PLATFORMVIEW Viewpoint
      {description "CAMARA GIRA"
       position 0 1 1 }
]}
]}

Group {
  children [
     DEF norm Viewpoint {
       position 0 0 15
       description "Normal Entry"}
     DEF aerial Viewpoint {
       position 0 15 0
       orientation 1 0 0 -1.57
       description "Aerial View"}
     DEF headcam Viewpoint {
       position 175 0 30
            orientation 0 1 0 1.57
       description "HeadCam"}
     DEF aerial Viewpoint {
       position 0 0 -10
       orientation 0 1 0 3.14
       description "Aerial View1"}
  ]
}
Background {skyColor 0 1 1}
DirectionalLight
{ direction -0.33104 -0.74485 -0.57932 color 1 1 1 intensity 1 on TRUE }

####  DIAGRAMA DE RADIACION  ####
Shape
     {appearance Appearance
      { material Material
        { specularColor 1 1 0.5
         diffuseColor 0.5 0 1
              ambientIntensity 0
              transparency
```

Programa en Visual Basic.

```
Private Sub cmdMostrar_Click()

 Dim x(160, 200), y(160, 200), z(160, 200) 'Definición de las 3 matrices para cada coordenada.
Dim pi As Double
pi = 4 * Atn(1)
Dim LongAnt As Currency

resol1 = 75                    'En el giro de phi...2*pi, resolucion de 150
resol2 = 190                   'En el giro de teta...pi, resolución de 190
r = 5
k = 0                          'Se inicia contador de filas a 0.
LongAnt = hsbLongitudAnt.Value / 20  'LongAnt es la relación entre la longitud del dipolo y la longitud de onda.

      For phi = 0 To 2.01 * pi Step pi / resol1  '2.01 en lugar de 2 porque cuando el contador llega a 2pi, no entra en el
lazo.
         k = k + 1               'Contador de filas de las 3 matrices.
         j = 1                   'Se inicia contador de columnas a 1.
         For teta = 0 To pi + 0.01 Step pi / resol2 'Se le suma 0.01 a pi porque cuando el contador llega a pi, no entra en
el lazo.
           If teta = 0 Then
             y(k, j) = 0         'Hacemos este lazo porque la función rp..
             x(k, j) = 0         'se hace indeterminado cuando teta es....
             z(k, j) = 0         'igual a cero.
           End If
           If teta > 0 Then
             If optSinInterf.Value = True Then  'Si es elegido el dipolo sin intereferencia.
               rp11 = Cos(pi * LongAnt * Cos(teta)) - Cos(pi * LongAnt)
               rp = CCur(r * rp11 / Sin(teta))
             End If
             If optInterfDiedro.Value = True Then 'Si es elegido el dipolo con intereferencia del diedro.
               rp1 = Cos(pi * LongAnt * Cos(teta)) - Cos(pi * LongAnt)
               'rp1 = Cos(pi * Cos(teta) / 2)
               rp2 = Cos(Sin(teta) * Cos(phi)) - Cos(Sin(teta) * Sin(phi))
               rp = CCur(r * rp1 * rp2 / Sin(teta))
             End If
               y(k, j) = CCur(rp * Cos(teta)) 'Se ingresa la coordenada y a la matriz y().
               xi = rp * Sin(teta)          'xi es la proyección de rp en el plano x-z.
               x(k, j) = CCur(xi * Sin(phi))  'Se ingresa la coordena x a la matriz x(), según el giro de phi.
               z(k, j) = CCur(xi * Cos(phi))  'Se ingresa la coordena z a la matriz z(), según el giro de phi.
           End If
               j = j + 1                  'Contador de columnas de las 3 matrices.
         Next
      Next

Open "c:\Tesis_dipolo\Lobulo1.txt" For Input As #1
Open "c:\Tesis_dipolo\Lobulo1.wrl" For Output As #2

   Do While Not EOF(1) '??????
      Input #1, slin
      Print #2, slin
   Loop

Close #1
'   ############ CONTINUACION DE INDEXEFACE ########
transparency = hsbTransparency.Value / 10
slin = CStr(hsbTransparency.Value / 10)
Print #2, slin
slin = " shininess 0.8 }}"
Print #2, slin
slin = "geometry IndexedFaceSet{ ccw FALSE creaseAngle 0 coord Coordinate { point ["
Print #2, slin

'  #######   SE LLENA CAMPO point      #######
   For i = 1 To resol1 * 2 + 1
```

```
        For j = 1 To resol2 + 1
            slin = CStr(x(i, j)) + " " + CStr(y(i, j)) + " " + CStr(z(i, j)) + ","
            Print #2, slin
            punto = punto + 1
        Next
    Next

slin = "]  } solid FALSE coordIndex ["
Print #2, slin
'    #########  SE ENLAZAN LOS DIFERENCIALES DE AREAS   ########
  i = 0
  Do Until i = punto - 1 - resol2 - 1 '31 se mantiene
        slin = CStr(i) + "," + CStr(i + 1) + "," + CStr(i + resol2 + 2) + "," + CStr(i + resol2 + 1) + ",-1,"
        Print #2, slin
        i = i + 1
  Loop

slin = "] creaseAngle 2 }  } "
Print #2, slin

'    ######## LINEA   ######
slin = "Shape {appearance Appearance {material Material {diffuseColor 0 0 0} }"
Print #2, slin
slin = " geometry IndexedLineSet {      coord Coordinate {  point ["
Print #2, slin

For i = 1 To resol1 * 2 + 1
        For j = 1 To resol2 + 1
            slin = CStr(x(i, j)) + " " + CStr(y(i, j)) + " " + CStr(z(i, j)) + ","
            Print #2, slin
            punto = punto + 1
        Next
    Next

slin = "]}coordIndex ["
Print #2, slin

 i = 0
  Do Until i = punto - 1 - resol2 - 1 '31 se mantiene
        slin = CStr(i) + "," + CStr(i + 1) + "," + CStr(i + resol2 + 2) + "," + CStr(i + resol2 + 1) + ",-1,"
        Print #2, slin
        i = i + 1
  Loop

slin = " ]  }}"
Print #2, slin

'    ######## INTERPOLADOR   ######
slin = "DEF ORIINT OrientationInterpolator {key [0, 0.25, 0.5, 0.75, 1]keyValue [    0 1 0 0, "
Print #2, slin
slin = "0 1 0 1.57, 0 1 0 3.14,0 1 0 4.71, 0 1 0 6.28 ]}"
Print #2, slin
' ######## ROUTE   ########
slin = "ROUTE TIMER.fraction_changed TO ORIINT.set_fraction ROUTE ORIINT.value_changed TO
ORITGT.set_rotation"
Print #2, slin



Close #2
 Form1.Show vbModal
End Sub
```

REFERENCIAS.

[1] Belotserkovski "Fundamentos de Antenas" Marcombo, S.A., Barcelona, España, (1983).

[2] García Domínguez, Armando "Calculo de Antenas" Segunda Edición, Marcombo, S.A., Barcelona, España, (1995).

[3] Jordan, Edward "Ondas Electromagnéticas y Sistemas Radiantes" Segunda Edición, Paraninfo, Madrid, España, (1978).

[4] Cardama / Jofre / Rius / Romeu / Blanch, Alfaomega "Antenas", (2000).

[5] Gurewich, Ori "Aprendiendo Visual Basic 5 en 21 días". Prentice Hall.

[6] "Floppy's VRML97 Tutorial"
http://web3d.vapourtech.com
Visitado por ultima vez 29 Marzo 2004.

[7]Diedro 90 grados
http://www.upv.es/antenas/Documentos_PDF/Problemas/Tema_5/dipolo_diedro_9pdf

[8]Diseño de un arreglo radiointerferométrico de antenas Yagi-Uda con centro de ventana espectral en 400 MHz
http://www.scielo.org.ve/scielo.php?pid=S0254-07702003000100003&script=sci_arttext
Visitado por ultima vez Oct 2004.

[9] J.L. Ramos Quirarte, M.J. Martinez Silva y M.S. Ruiz Palacios: Software para el Calculo de Patrones de Radiacin de Arreglos Lineales de Antenas
http://exa.unne.edu.ar/depar/areas/informatica/SistemasOperativos/cie2001109.pdf
Visitado por ultima vez 15 Oct 2004.

[10]J.L. Ramos Quirarte, M.J. Martinez Silva y M.S. Ruiz Palacios: Arreglos Planos: La cruz Mills.
http://exa.unne.edu.ar/depar/areas/informatica/SistemasOperativos/cie200286.pdf
Visitado por ultima vez 15 Oct 2004.

[11] Miguel Ferrando, Alejandro Valero: Agrupaciones Planas
http://www.upv.es/antenas/Documentos_PDF/Notas_clase/Agrupaciones_planas.pdf
Visitado por ultima vez 22 Oct 2004.